Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA GATEWAY RCS-e VIDEO SHARE

Autor: Jesús Avilés Avilés

Tutor: Antonio de la Oliva Delgado

Leganés, Julio de 2012

Agradecimientos

De manera especial, a mis padres, por el apoyo y la confianza que siempre me dieron en los momentos de flaqueza, sin duda, sin ellos no hubiera llegado hasta aquí. El logro que supone la consecución de mis estudios universitarios también es el suyo.

A mis hermanas, por el apoyo que siempre dieron al "Chico", y a Vanesa, por sus ánimos que tan importantes han sido en esta última etapa.

En lo profesional, a mis compañeros de la empresa Solaiemes, por sus consejos y ayuda, especialmente a Luis, que me dirigió durante este proyecto con toda la paciencia del mundo. Por último, a Antonio, por su plena colaboración a la hora de dar forma a este documento.

Resumen

Rich Communication Suite (RCS) es una iniciativa de los operadores y fabricantes de telefonía móvil, mediante la que se trata de redefinir una serie de servicios IMS ya existentes, como son IM/chat, transferencia de ficheros o compartición de video e imágenes. Su objetivo es buscar homogeneidad de mecanismos de funcionamiento que permita una alta interoperabilidad, factor clave para que estos servicios sean ampliamente aceptados por los usuarios.

Entre los servicios recogidos por RCS, se encuentra Video Share, servicio que permite a los usuarios compartir video en tiempo real, ya sea éste pregrabado o capturado en vivo por las cámaras de sus terminales. RCS-e, nombre con el que proyecto recientemente ha visto la luz, integra el servicio VS junto con el resto de servicios en la agenda del teléfono. Los usuarios mediante un cliente embebido en su terrminal, accederán a su agenda para seleccionar el contacto con el que desean establecer una sesión VS.

En este proyecto se presenta una propuesta, mediante la cual, el servicio VS, en principio definidos por RCS-e para la compartición de video usuario a usuario, es abierto a desarrolladores (terceras partes), para la creación de aplicaciones y casos de uso donde el servicio VS pueda ser integrado con soluciones web y servicios de contenidos, de forma que se vea ampliada la oferta para el usurio.

El sistema RCS-e VS Gateway consiste en una pasarela o gateway entre RCS-e y aplicaciones VS. El sistema basara su funcionamiento en la creación y gestion de clientes virtuales con capacidad VS, mediante los cuales se podrá establecer sesiones de video con clientes RCS-e embebidos en terminales móviles, a través del core IMS/RCS-e del operador, empleando la interfaz usuario-red (UNI). Estos clientes virtuales serán asignados a aplicaciones externas, que podrán acceder a la funcionalidad VS ya implementada sobre ellos, a través de llamadas a un API de Web Services.

Este proyecto tiene como objetivo abordar el diseño y la implementación del sistema RCS-e VS Gateway. Adicionalmente se implementa un caso de uso concreto, consistente en una aplicación Videoblog, con el objetivo de mostrar las capacidades y posiblidades del sistema.

Palabras claves: Rich Communication Suite (RCS), Video Share (VS), Gateway, SIP, RTP, Web Services, Videoblog.

Índice general

Índice figuras

Índice tablas

Índice Extractos

Capítulo 1

Introducción.

En este primer capítulo se ofrece una presentación del proyecto realizado. En primer lugar se expone el contexto tecnológico sobre el que se ha trabajado. A continuación se hablará acerca de los motivos que ha llevado a la realización de este trabajo, exponiendo el problema a resolver y dando una descripción detallada de la solución que se propone y la aportación que puede ofrecer. Después se enumeran los objetivos que implica el desarrollo de la solución propuesta y finalmente se presenta la estructura detallada de este documento.

1.1 RCS-e y servicio Video Share

Con la intención de llevar los servicios multimedia, ya ampliamente extendidos en Internet, al mundo de la telefonía móvil, surgió IMS (IP Multimedia Subsystem). En sí, IMS no define como son las aplicaciones finales de las que dispondrá el usuario, solamente se encarga de dar la infraestructura y el soporte necesarios para los provisión de esos servicios. Esto ha dado lugar al desarrollo de muy diversas aplicaciones, con funcionamiento e interfaces de usuario muy dispares, esta heterogeneidad además se ha visto alimentada por la diversidad de gamas de los terminales y por la falta de acuerdos entre operadores. El resultado es una falta de interoperabilidad que representa uno de las principales causas de que los nuevos servicios multimedia no hayan sido aceptados en la medida esperada por los consumidores.

Rich Communication Suite (RCS) trata de abordar este problema redefiniendo un conjunto de servicios ya existentes, como son el IM/chat, la transferencia de ficheros y la compartición de video e imágenes desde la perspectiva de la aplicación final, buscando homogeneidad de mecanismos de funcionamiento que permitan una amplia interoperabilidad. Mediante clientes RCS embebidos en terminales de media y alta gama, los servicios son integrados en torno a la agenda de contactos telefónicos, dando lugar a una interfaz de usuario común a todos los servicios, facilitando y enriqueciendo así la experiencia del usuario.

Uno de los servicios multimedia recogidos por RCS es el Video Share. Video Share es un servicio definido en un principio para la compartición de video durante una llamada de voz, es decir, un usuario puede capturar o transmitir video (ya sea en vivo mediante la cámara del móvil o pregrabado) de/a otro usuario en tiempo real durante una llamada de voz previamente establecida. El servicio de Video Share supone un primer paso para abrir con éxito las comunicaciones telefónicas al dominio visual y proporcionar a los operadores móviles una oportunidad para ampliar la gama de servicios ofrecidos.

En el capítulo 2, dedicado al estado del arte, se explica en qué consiste el proyecto RCS, y se da en detalle una descripción funcional y técnica del servicio Video Share en el entorno RCS. Aquí a modo introductorio diremos que la versión actual de RCS, que manejan operadores y fabricantes, conocida como RCS-e (RCS enhanced), define Video Share utilizando como base la especificación IR.74, publicada por la GSMA (Global System for Mobile Comunications Association) para la implementación y el desarrollo de este servicio.

Bajo esta especificación, Video Share viene dado como servicio punto a punto que utiliza el core IMS para la transmisión de vídeo entre terminales móviles. Las sesiones son establecidas utilizando el protocolo Sesion Initiation Protocol (SIP), y la transmisión de paquetes de video se realiza mediante el protocolo Real-Time Transport Protocol (RTP). En principio el modelo utilizado es P2P, siendo las aplicaciones implementadas en los terminales, por lo que no es necesario añadir servidores de aplicaciones adicionales.

1.2 Motivación

Mediante RCS-e un usuario con un terminal móvil, que cuente con un cliente RCS-e embebido, puede buscar en su agenda de teléfono contactos que disponga también del cliente, visualizando al instante mediante iconos cuales son las capacidades RCS-e que soporta (IM/chat, transferencia de ficheros, compartición de imágenes o compartición de video). Seleccionando cualquiera de esos servicios disponibles se establece la sesión multimedia correspondiente con el mismo.

Como se puede ver, RCS-e en un principio está orientado a la prestación de servicios IMS básicos para la comunicación entre usuarios. En su objetivo de explotar las capacidades del sistema IMS, el siguiente paso debería ser llevar estos servicios hacia el desarrollo de aplicaciones que supongan un valor añadido para el usuario. Esto implica una extensión de la comunicación usuario-usuario a usuario-contenidos y la integración

de la funcionalidad de RCS con los servicios y las soluciones web de mayor éxito en Internet.

Para el caso de Video Share esta evolución ha sido recogida por la especificación IR.84 de la GSMA, explicada más abajo, en el capítulo dedicado al estado del arte. Esta propuesta plantea un escenario donde se hace necesario la incorporación adicional de servidores de aplicaciones sobre el core IMS, al igual que el de otras entidades IMS necesarias para la manipulación y gestión del media, como son los Multimedia Resource Function. También se incluyen otros bloques no pertenecientes a la arquitectura IMS como son un portal Web y un repositorio de media para las sesiones usuario-contenido. Este nuevo escenario permite ampliar la funcionalidad del servicio e implementar aplicaciones de mayor valor a cambio de una mayor complejidad de la arquitectura, dejando atrás el modelo P2P con el que había sido concebido el servicio Video Share inicialmente. En cuanto a su adopción en RCS-e, esta propuesta supondría profundos cambios ya no sólo a nivel de infraestructura en el core IMS/RCS sino que también sería necesario adaptar los clientes RCS-e de los terminales para que éstos puedan interactuar con los nuevos servidores de aplicaciones VS. Esta restructuración de mecanismos de funcionamiento puede suponer un obstáculo respecto a la premisa de interoperabilidad que persigue el proyecto RCS-e.

En este proyecto se trata de dar una solución alternativa, mediante la cual se puedan desplegar servicios de valor añadido basados en Video Share, sin introducir cambios en la infraestructura del operador (core IMS/RCS) y manteniendo los mecanismos de funcionamiento, adaptándose así a los dispuesto por RCS-e y la especificación IR.74 para la compartición de video. La solución consiste en una pasarela o gateway entre RCS-e y aplicaciones VS empleando la interfaz usuario-red (UNI).

El sistema diseñado e implementado en el presente proyecto permite crear y gestionar clientes virtuales con capacidad VS, bajo los requerimientos RCS-e, de forma que puedan establecer sesiones de video a través del core de la red, utilizando el interfaz usuario a red UNI de la misma forma que lo hacen los cliente RCS-e embebidos en los terminales de usuarios. Esos clientes virtuales serán utilizados por aplicaciones externas que requieran de la funcionalidad de VS para un determinado propósito. El sistema asigna clientes virtuales a las aplicaciones, de forma que detrás de un cliente virtual en realidad no hay un usuario sino una aplicación. A través de estos clientes la comunicación e interacción entre la aplicación y el usuario se hace del mismo modo a como lo hacen comúnmente dos usuarios RCS-e compartiendo video.

Los clientes virtuales como cualquier otro cliente con capacidad videoshare, emplean SIP para el establecimiento de sesión, RTP para el transporte de paquetes de video y RTCP para el control del transporte, adicionalmente también incluirán soporte RTSP (Real Time Streaming Protocol) y RTMP (Real Time Messaging Protocol) para la publicación y captura de flujo de video en vivo

La plataforma tiene como objetivo dar soporte a desarrolladores de aplicaciones (terceras partes), que quieran implementar algún servicio o aplicación con funcionalidad VS, sin la necesidad de adquirir los conocimientos que requiere la infraestructura del operador (arquitectura IMS, protocolos SIP, RTP, etc.) ni someterse a los largos procesos de interoperabilidad necesarios en IMS. El sistema proporciona un

API mediante el cual los desarrolladores software podrán acceder a funcionalidad ya implementadas sobre los clientes virtuales.

Respecto al usuario final hay que decir que la solución presentada tiene el valor añadido de que no requiere cambios en los terminales, ni instalar nuevo software para cada nueva aplicación. El acceso a las aplicaciones se hace a través de la agenda de contactos, al igual que se haría para comunicarse con otro usuario RCS-e, manteniéndose así el principio de interfaz único en torno a la agenda de contactos previsto por RCS. Las aplicaciones como si se tratase de cualquier otro contacto RCS-e cuentan con un identificador único SIP-URI o TEL-URI (identificadores empleados en IMS) asignado al cliente virtual al que esta ligados, mediante el cual están accesibles

Con este sistema se abre enormemente el abanico de posibilidades del servicio. Los desarrolladores pueden crear casos de uso como pueda ser el de la transmisión de video en tiempo real de móvil a web, de manera que tus amigos puedan ver lo que tú estás viendo desde una red social. Se pueden crear videoblogs personales donde publicar tus experiencias a través del video capturado desde tu móvil. Las posibilidades en torno a los UGC (User Generated Contents) son numerosas. Desde el punto de vista empresarial, ejemplo de uso sería la difusión publicitaria, pudiéndose informar a los usuarios de ofertas y otras cuestiones a través de videos que se hacen llegar hasta sus móviles (Customer Relationship Management) y con respecto a los empleados, por ejemplo, éstos podrían mandar videos a través de sus móviles a modo de informes de campo en tiempo real para toma de decisiones de forma remota, etc.

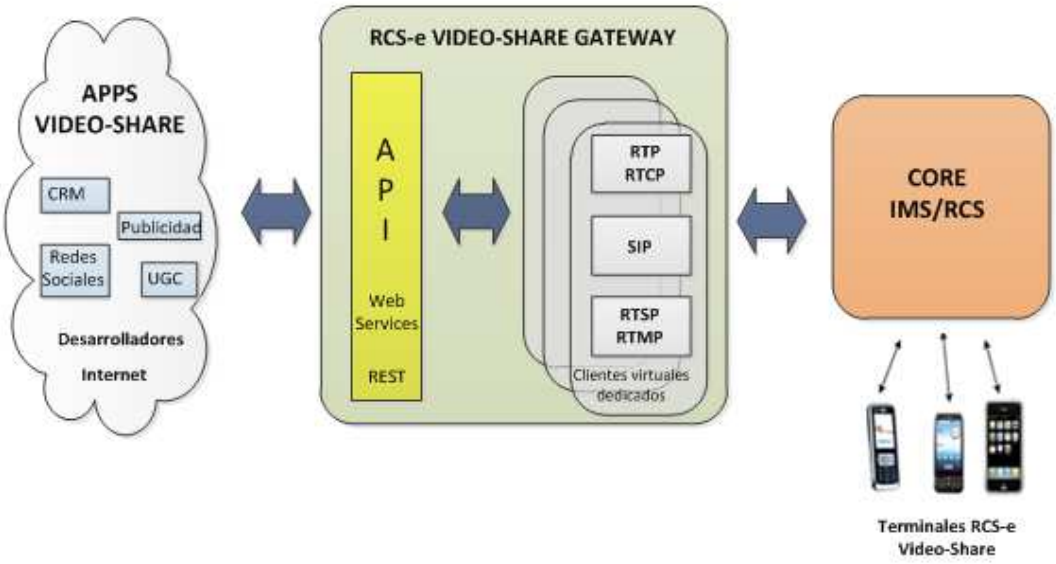


Figura 1.Propuesta del sistema RCS-e Video Share Gateway

1.3 Objetivos

Como ya se ha descrito en el apartado anterior, el objetivo principal del proyecto consiste en diseñar e implementar un sistema pasarela entre la actual implementación del servicio VS en RCS-e y nuevas aplicaciones fundamentadas en soluciones web y repositorios de video. A continuación se enumeran los objetivos parciales que se han perseguido a la hora de desarrollar este proyecto:

-Estudio y análisis del proyecto RCS-e y del servicio Video Share: en qué consisten, protocolos y tecnologías que emplean, situación actual y líneas futuras de trabajo.

-Estudio de la plataforma de programación JEE, elegida para el desarrollo en lenguaje java del software del sistema. Estudio y valoración de las herramientas y API´s que ofrece, para el desarrollo de las distintas funcionalidades del sistema: Enterprise JavaBeans, Servicios Web, Servlets, RMI, JDBC, etc.

-Implementación de un servidor de control y señalización SIP, para el establecimiento y control de las sesiones Video Share. Será implementada sobre la plataforma SIP Servlets Mobicents, de uso muy extendido en telecomunicaciones, que ofrece un API en java para atender peticiones SIP mediantes Servlets.

-Desarrollo de un sistema para la creación y administración de múltiples clientes virtuales con capacidad Video Share. Diseño y gestión dinámica de una base de datos con la información y el estado de clientes-aplicaciones y sesiones. El sistema será implementado sobre JBoss, un servidor de aplicaciones JEE, que ofrece la escalabilidad y flexibilidad requerida por el sistema.

-Implementación de un servidor de video que ofrezca soporte RTP/RTCP para envío y recepción de video bajo los requerimientos RCS-e. Integración y adaptación de un servidor de video en vivo, Liveserve de la empresa Solaiemes, con funcionalidad cliente/servidor para la publicación y captura de flujos de video en vivo hacia o desde las aplicaciones web mediante los protocolos RTSP y RTMP.

-Gestión por parte del sistema de un repositorio externo de contenidos multimedia para dar soporte a aplicaciones basadas en comunicación usuario-contenido.

-Diseño y desarrollo de un API de entrada al sistema constituido por acciones y notificaciones que permita la comunicación ente aplicación y Gateway en ambos sentidos. El API será implementado mediante Web Services, de forma que los desarrolladores de aplicaciones puedan hacer uso de las funcionalidades del Gateway desde sistemas remotos a través de Internet.

-Aunque el sistema tiene las pretensiones iniciales de ser un prototipo, se tiene como objetivo la escalabilidad del sistema y sobre todo la interoperabilidad del mismo. Se harán pruebas con clientes RCS-e de diferentes desarrolladores y con cores IMS/RCS-e de diferentes fabricantes y operadores de telecomunicaciones, siempre bajo la limitación que supone la fase de desarrollo en la que se encuentra el proyecto RCS-e actualmente.

-Implementación de un caso de uso del API del Gateway, a modo de ejemplo, que sirva de referencia para los desarrolladores de aplicaciones. La aplicación consistirá en un sencillo Videoblog, utilizando el API que ofrece Google para Blogger y por supuesto el API del Gateway de Video Share.

1.4 Estructura de la memoria.

El presente documento se encuentra estructurado en diferentes capítulos. A continuación se describe de manera breve el contenido de cada uno:

-Capítulo 1 – Introducción: En este capítulo se establece el marco de estudio del proyecto, presentando la necesidad de la implementación desarrollada así como los objetivos que implica.

-Capítulo 2 – Estado del arte: Describe el entorno tecnológico en el que el proyecto se va a desarrollar, en este caso se trata del servicio Video Share en el entorno RCS-e. Se describen la arquitectura soporte, los protocolos participantes y la situación actual de su implementación a modo de punto de partida.

-Capítulo 3 – Diseño del sistema: En primer lugar se describe la funcionalidad general del sistema, cuales son los casos de uso a implementar y se especifica los requisitos técnicos del sistema. Posteriormente se presenta la arquitectura del sistema y se explican la funcionalidad de cada uno de los módulos que la componen.

-Capítulo 4 – Implementación del sistema: Una vez conocida la funcionalidad del sistema y los elementos que lo conforman, se procede a explicar cómo se han implementado. Primero se describen las herramientas y medios empleados para cada módulo y posteriormente se expone el modelado software implementado.

-Capitulo 5 – Uso del sistema: Videoblog. La funcionalidad final del sistema es dar soporte a desarrolladores de aplicaciones. Aquí se expone como utilizar el API, a través de un sencillo ejemplo de aplicación consistente en un videoblog.

-Capitulo 6 – Planificación y presupuesto. Se detallan las diferentes fases del desarrollo y se realiza el presupuesto del proyecto.

-Capítulo 7 – Se resumen la principales conclusiones del proyecto realizado y se enumeran una serie de posibles líneas de trabajo futuras.

Capítulo 2

Estado del arte.

Este capítulo tiene como objetivo dar una descripción funcional del servicio Video-Share en el entorno RCS y tratar los aspectos técnicos más relevantes del mismo, de modo que sirva como base teórica y punto de partida para el desarrollo del sistema RCS-e Video Share Gateway.

Este estado del arte se puede dividir en tres grandes bloques según lo recogido en los distintos apartados. Para empezar, en los dos primeros apartados se dará una perspectiva general acerca del proyecto RCS y del servicio de Video Share. Los siguientes tres apartados tendrán como objetivo presentar la tecnología base sobre las que se sustenta Video Share, siendo IMS la arquitectura soporte, SIP el protocolo de establecimiento de sesión y RTP el protocolo empleado para el transporte del video. Finalmente, con la base que aporta todo lo anterior, se describirá de forma detallada el servicio VS, primero dándole un enfoque más técnico y luego poniendo mayor atención sobre los aspectos funcionales más relevante, terminando con un último punto donde se darán las bases de lo que debería ser la evolución del servicio en el futuro**.**

2.1 Rich Communication Suite

2.1.1 ¿Qué es RCS?

RCS [1] es un proyecto promovido por la GSMA [2], enfocado a la explotación del sistema IP Multimedia Subsystem (IMS) [ver sección 2.3] para la provisión de servicios de telefonía móvil mejorados o enriquecidos respecto a los ya tradicionalmente ofrecidos al cliente y de una forma totalmente interoperable.

Los componentes fundamentales de RCS son la mensajería mejorada y la llamada enriquecida. En el primer caso, el usuario dispondrá de una alta variedad de opciones de mensajería, incluyendo chat, historial de mensajes y transferencia de ficheros. Con la llamada enriquecida, durante una llamada de voz, se podrán compartir contenidos multimedia, tales como imágenes o videos, ya sea estos últimos en vivo o almacenados.

En realidad RCS no define nuevos servicios, su propuesta es la de una eficiente agrupación y perfilado de servicios estandarizados, ya existentes, que supongan una oferta realmente atractiva para el usuario final. En esta línea juega un papel fundamental el tercer gran pilar de RCS, que es conocido como agenda de teléfono mejorada. Todos los servicios estarán integrados de cara al usuario en la agenda del móvil, donde será visible las capacidades de cada uno de los contactos en cuanto a servicios que pueden soportar. De esta forma el cliente estará informado en todo momento de las formas en las que puede contactar con otro usuario y podrá elegir entre ellas. Adicionalmente a lo anterior, la agenda enriquecida podrá soportar presencia, es decir, visibilidad para el usuario final de información acerca de la presencia social de sus contactos a través de la agenda. La información de presencia se podrá apoyar en algunos de los siguientes atributos: indicadores de disponibilidad (ocupado, temporalmente ocupado, disponible, etc.), icono o fotografía representativa que muestra el contacto de cara a los demás, enlaces favoritos, nota de texto libre incluyendo emoticonos, fecha y hora de la actualización del perfil, etc.

2.1.2 Evolución de RCS. Especificaciones

El proyecto RCS, trata de poner de acuerdo a fabricantes de terminales e infraestructura, desarrolladores de aplicaciones, y operadores de telecomunicaciones, en su objetivo de extender los servicios multimedia y lograr una amplia interoperabilidad. El proyecto es recogido en una serie de especificaciones, que deberían servir de guía a todas las partes implicadas. La especificación original, publicada en 2008, ha ido evolucionando y ampliándose con el tiempo dando lugar a cuatro releases. Cada release a su vez cuenta con una serie de documentos donde se recogen la definición de servicios, la descripción funcional y la realización técnica. Aquí se listan las características de los servicios abordadas por cada release [1]:

Release 1

- Agenda de contactos mejorada:
  - Presencial social.
  - Información de capacidad de servicios
  - Lista negra de contactos
  - Soporte de contenidos enriquecidos
  - Backup/sincronización con registro de direcciones en red.

-Compartición de contenidos durante una llamada de voz tradicional de conmutación de circuitos (Circuit Switching).

-Transferencia de ficheros durante una llamada de voz o en conversación de mensajes.

-Mensajería mejorada:
- Mensajería conversacional sobre el dispositivo.
- Composición unificada para SMS/MMS.
- Vista circular de mensajes SMS/MMS
- Servicio chat.
- 1 a 1 y 1 a muchos.
- Basado en sesiones de mensajería basadas en OMA IM SIMPLE 1.0

Release 2

-Acceso a RCS mediante dispositivos de Banda Ancha (PC, tabletas):

-Entorno multi-dispositivo, una suscripción válida para varios dispositivos:
- Uno primario y dos dispositivos secundarios.
- El cliente primario debe tener capacidad de acceso celular.
- Interacción de todos los dispositivos sobre la misma agenda de contactos.

-Registro de direcciones en red
- El operador móvil administra y mantiene.
- Incluye todos los dispositivos pertenecientes a un usuario.
- Control y sincronización con la agenda de teléfonos mejorada de usuarios.
- Aprovisionamiento y configuración de clientes/dispositivos RCS
- Configuración de dispositivos RCS transparente a consumidores.

Release 3

-Dispositivo de banda ancha puede ser primarios.

-Mejora en la compartición de contenidos.
- Compartición de contenido sin llamada de voz.
- Información de geo-localización en texto y/o mapas personalizados.
- Etiquetas URL.

- Capacidad para mostrar información de servicios soportados por los contactos de la agenda, incluso si no hay relación de presencia social establecida.

-Mensajería mejorada
- Capacidad para los dispositivos de banda ancha para enviar y recibir SMS/MMS.
- Lista de invitados participantes/receptores para comunicaciones en grupo.

-Servicios de valor añadido en red. Enriquecimiento de contenidos y chat con procesamiento del media (p.e. traducción de idiomas).

-Aprovisionamiento y configuración transparente de dispositvos/clientes RCS.

Release 4

-Incluye todas las release listadas anteriormente.

-Soportado por Long Term Evolution (LTE):
- Tamaño de mensaje de texto.
- Múltiples recipientes de mensajes de texto.
- Capacidades mejoradas de MMS
- Manipulación de imágenes compartidas
- Contactos VIP
- Video Share con pausa y reanudamiento.

2.1.3 RCS-e

RCS-e (enhanced) [3], es la última iniciativa tomada en torno al proyecto RCS (Febrero de 2011), con la intención de acelerar su salida al mercado. Se trata de una especificación basada en las Release 1 y 2 de RCS [4] [5] con ciertas modificaciones para lograr mayor eficiencia en el establecimiento de la interoperabilidad y simplificar la propuesta al consumidor. Estas modificaciones están basadas en los resultados de diferentes programas de prueba que se han llevado a cabo en distintos países.

RCS-e se define como una simple e interoperable evolución para la voz y el texto, que permitirá a los consumidores enviar mensajes instantáneos, video, chat e intercambio de ficheros en tiempo real. Todas estas funciones incluidas en la agenda de contactos del móvil.

La agenda de contactos mejorada en RCS-e deja fuera, de manera provisional, los mecanismos previstos para mostrar la presencia social de los contactos [6], centrándose sólo en mostrar las capacidades en tiempo real, es decir los servicios soportados por un contacto en cada instante, independientemente de los servicios con los que se haya sido registrado en la red.

Para tal fin, RCS-e define un mecanismo de intercambio de capacidades basado en el envío de mensajes SIP OPTIONS punto a punto entre usuarios, y una serie de etiquetas identificativas de servicio que irán en esos mensajes. Un mecanismo de notificación de eventos y otro basado en pooling para el envío periódico de OPTIONS a todos los contactos, permitirá tener conocimiento en tiempo real de las capacidades de éstos [ver sección 2.4 y 2.7.5].

Como ya se ha comentado, RCS no define nuevos servicios, se basan en servicios ya existentes definidos por sus propias especificaciones. En la siguiente figura se muestra los servicios que engloba RCS-e y las especificaciones o mecanismos sobre los que se basan:

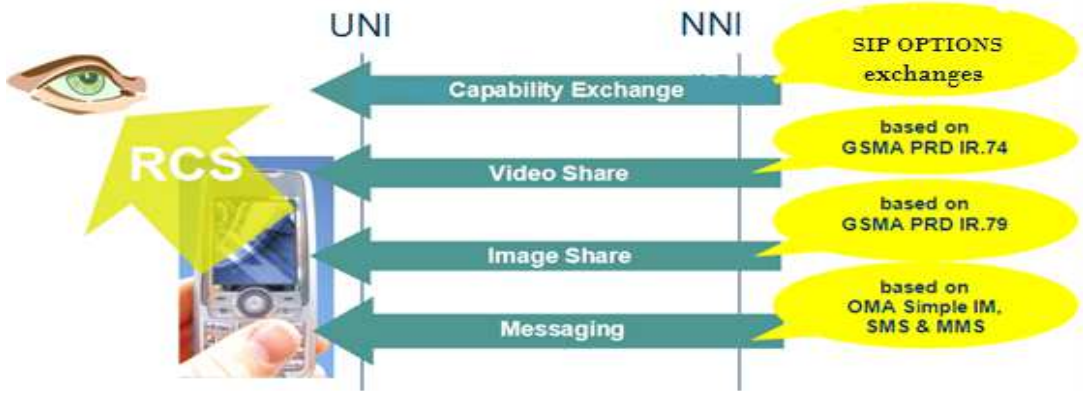


Figura 2: Servicios y especificaciones de RCS-e.

2.2 Definición de servicio Video Share

Video Share [7] es un servicio definido en un principio para la compartición de video durante una llamada de voz, es decir, un usuario puede capturar o transmitir video (ya sea en vivo o pregrabado) de/a otro usuario en tiempo real durante una llamada de voz previamente establecida. La principal diferencia con respecto al servicio de video-llamada es que no incorpora la sincronización con el audio, siendo la conexión totalmente independiente. Video Share es un proyecto de la GSMA, cuya implementación y desarrollo está dividida en dos fases, recogidas éstas en sendas especificaciones técnicas [8] [9].

En la fase 1 el servicio está definido como uni-direccional, se establece un flujo de video en tiempo real bajo una conexión PS, de un usuario a otro en un sólo sentido, pudiéndose dar cuando una llamada CS de voz ha sido ya establecida entre ambos, esto es debido a que en un principio está definido como un servicio complementario a la llamada de voz. Puede haber varias sesiones de video establecidas de forma secuencial, ya sean entrantes o salientes, pero en todo caso, independientes entre sí y bajo la restricción de la asociación de una llamada CS. Será soportado por cualquier dispositivo móvil, dispositivos de mano tradicionales o cualquier otro, con 3G habilitado. Los servicios avanzados asociados a la llamada de voz, tal como llamada en espera, conferencia, etc. no son aplicables al Video Share.

La principal diferencia de la fase 2 respecto a la fase 1 está en que la asociación entre llamada de voz y sesión de Video Share se rompe, aunque sigue siendo posible la compartición de video durante una llamada CS, ésta no es necesaria para establecer el flujo de video entre las partes. En la fase 2 se abre el servicio a nuevas posibilidades, como la asociación de VS a sesiones IM, la incorporación de información de presencia, VS punto a multipunto, y otras características de mejora. Para ello, a diferencia de la fase 1 donde el flujo de video tenía una clara orientación P2P entre usuarios finales, se incorpora servidores de aplicaciones de Video Share VS-AS, que permiten estas mejoras del servicio.

Video Share forma parte del conjunto de servicios RCS, más concretamente se incluye dentro de la denominada llamada enriquecida y de esta forma es recogida por las diferentes releases de RCS. Aunque es cierto que las releases 3 y 4 [1] ya incorporan el VS en su fase 2, la especificación RCS-e que actualmente manejan la industria, se centra en la primera fase. Por ello que tomaremos como guía en el presente estudio la especificación referente a la primera fase, dedicando un apartado extra para hablar de las novedades funcionales y técnica que supone la fase 2 del proyecto.

Video Share es un servicio punto a punto que utiliza el core IMS [ver sección 2.3] para transmitir paquetes de vídeo. Las sesiones son iniciadas utilizando el protocolo Sesion Initiation Protocol (SIP) [ver sección 2.4], y la transmisión del video se realiza mediante el protocolo Real-time Transport Protocol (RTP) [ver sección 2.5]. En principio el modelo utilizado es P2P, siendo las aplicaciones implementadas en los terminales, por lo que no es necesario añadir servidores de aplicaciones adicionales. El servicio debe ser interoperable entre operadoras con diferentes cores de IMS y entre usuarios con diferentes dispositivos 3G.

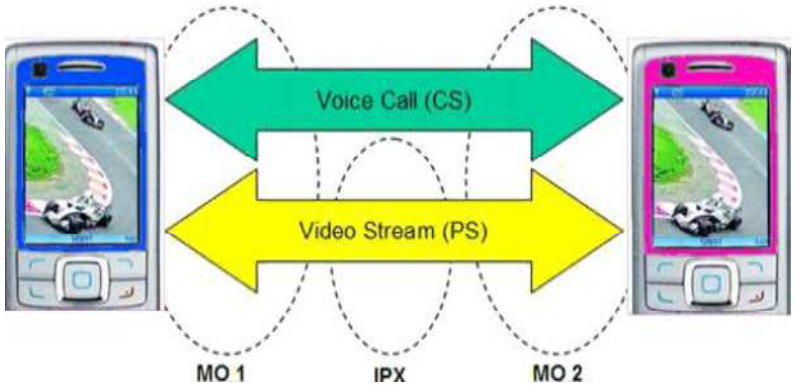


Figura 3: Conexiones a alto nivel de Video Share.

Los pasos básicos de una sesión de Video Share son los siguientes:

1. Establecimiento de llamada CS.
2. **Petición de capacidades.**
3. **Método de invitación de sesión (SIP).**
4. **Transmisión de video (RTP).**
5. **Finalización de sesión de video**

6. Finalización de llamada de voz.

A continuación, en los siguientes apartados se hace un recorrido muy breve sobre el sistema IMS y los protocolos participantes en una sesión de VS. Esto será fundamental para poder introducirnos en aspectos más técnicos del servicio y poder presentar y analizar las características más relevantes de su funcionamiento, de acuerdo a las especificaciones RCS-e [3] y Video Share fase 1 [8].

2.3. IMS

2.3.1 Definición y características de IMS

IP Multimedia Subsystem (IMS) [10] [11], es un sistema para el establecimiento y control de sesiones multimedia que trata de proporcionar un marco común o plataforma estandarizada para el desarrollo de nuevos servicios. Está basado en tecnología IP de conmutación de paquetes de datos, para tráfico de voz, video, audio e imágenes.

Usando IMS, los operadores y terceras partes podrán desarrollar fácilmente nuevos servicios que compartan un entorno común que facilite su integración. Algunos de estos servicios pueden ser: mensajería multimedia sobre IP, video-llamadas, video sharing, voz sobre IP, presencia, streaming, etc. IMS constituye el núcleo de la red de comunicaciones e implementa el plano de control de los servicios multimedia IP, constituyendo un nexo de unión entre diferentes tecnologías.

Algunas características básicas de IMS son:

-Está definido por un conjunto de especificaciones de 3GPP [12], y está basado en los estándares IETF [13].

-El plano de señalización y plano de control son independientes.

-Habilita mecanismos de seguridad como autenticación y autorización

-Proporciona esquemas de facturación por provisión de servicios.

-Soporta roaming.

-Provisiona mecanismos para la negociación de calidad de servicio QoS.

-Permite la interoperabilidad entre redes tradicionales de conmutación de circuitos e Internet.

IMS surge en un principio con el objetivo de hacer converger los dos grandes mundos de las comunicaciones, por un lado Internet que cuenta con una amplia variedad de servicios basados en protocolos muy comúnmente usados, y por el otro la telefonía móvil, cuyos servicios se centran en la voz pero que cuenta con un enorme número de suscriptores. Se trata de proporcionar servicios Internet como servicios multimedia,

donde el operador controla la negociación de la QoS, la autenticación de usuarios, la autorización para el acceso y la facturación.

En IMS los principales protocolos usados son SIP para el establecimiento y control de sesiones, y protocolos basados en DIAMETER [14] para la autenticación, autorización y contabilización (AAA) además de otros muchos protocolos comúnmente usados en Internet que han sido adaptados para los requerimientos del entorno móvil, el control por parte de los operadores y la interoperabilidad.

2.3.2 Arquitectura IMS

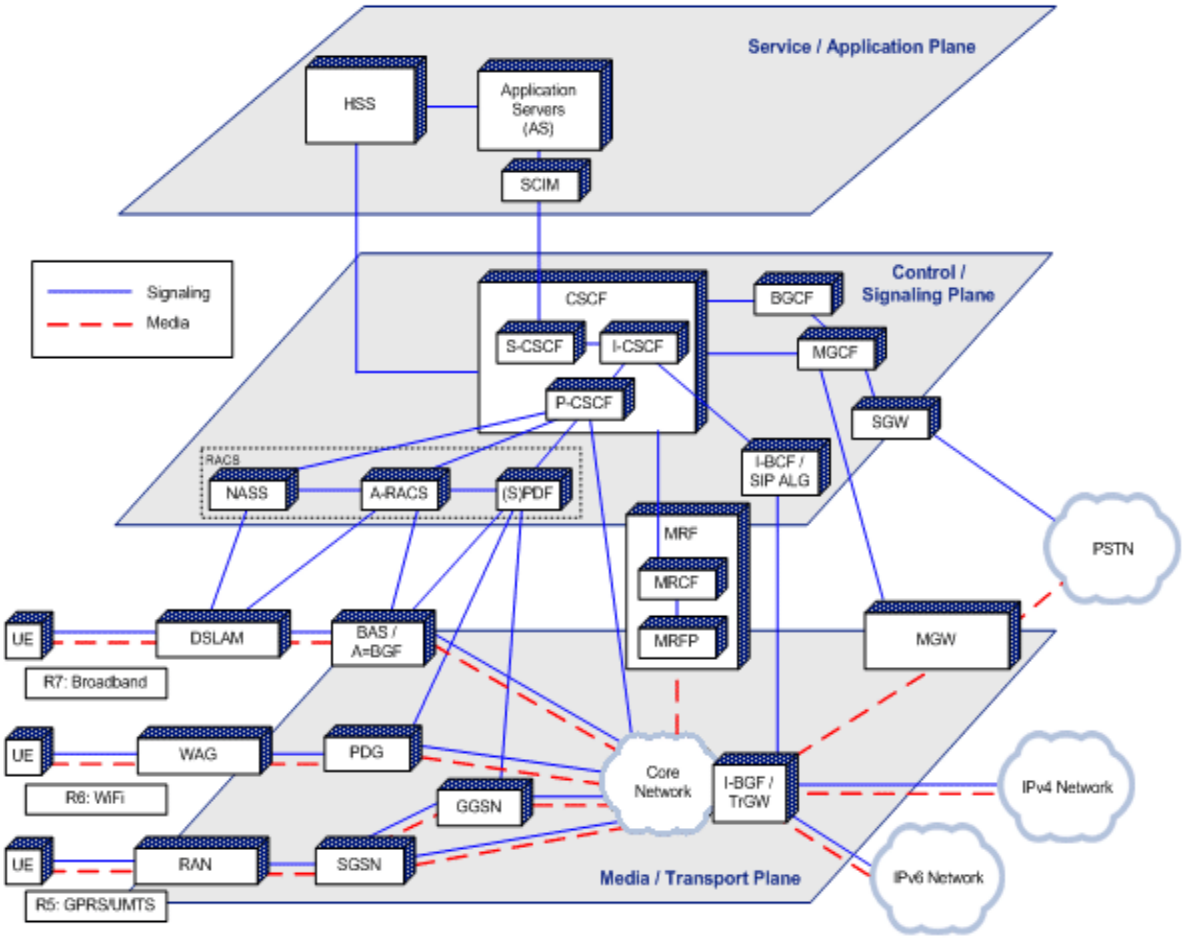


Figura 4: Arquitectura IMS.

- **CSCF. Call State Control Function. SIP Servers** [ver sección 5].

-**Proxy-CSCF**: Se trata del primer punto de contacto del usuario dentro de IMS. Todo el tráfico de señalización del usuario pasa por este nodo. Se encarga de procesar los mensajes que envía el usuario, validando la petición y reenviándola a los destinatarios.

-Interrogating-CSCF: Su función es la de determinar dentro de la red del operador, contra que S-CSCF el usuario se ha de registrar, recibe peticiones de registro de P-CSCF y manda consultas a HSS. Actúa como servidor proxy de SIP para establecimiento de sesiones. Es el punto de entrada de redes externas y su dirección se obtiene utilizando Domain Name System (DNS) [15].

-Serving-CSCF: Se puede considerar como el cerebro del IMS y se localiza en la red del operador. Realiza funciones de control de sesión y registro del usuario. Mientras un usuario se encuentra activo en la sesión, el S-CSCF mantiene el estado de la sesión e interactúa con otras plataformas para realizar funciones como el control de tarificación, o el inicio de servicios.

- **HSS. Home Subscribe Server.**

Es la entidad encargada de almacenar información de suscripción y el perfil de los usuarios, incluye la identidad de los usuarios e información de registro, además de parámetros de acceso y servicios. Esta información es utilizada en el establecimiento de sesiones para la autenticación y autorización de usuarios y para los mecanismos de roaming. Utiliza el protocolo DIAMETER [14] para la comunicación con I-CSCF y SIP/OSA AS. En una misma red puede haber más de un HSS, en este caso para localizar el HSS correspondiente a cada usuario se utiliza otra entidad llamada Subscription Location Function (SLF).

- **AS. Application Servers.**

Los servidores de aplicaciones se encuadran dentro de la arquitectura IMS en el plano de aplicación. Son nodos donde los servicios son implementados usando Java, SIP, servlets, SIP CGI, etc., dependiendo de los servicios, pueden actuar como proxies SIP, o de forma más compleja como User Agent (UA o B2BUA).

Interactúan con S-CSCF participando en las sesiones de control y con HSS para la actualización de los perfiles de usuario y para los cambios de estado de suscripción, adicionalmente puede hacer de interfaz con los usuarios para la configuración y gestión de servicios.

Los servicios ofrecidos no se limitan únicamente a servicios basados en el protocolo SIP, cabe la posibilidad de hacerlo siguiendo Customised Applications for Mobile network Enhanced Logic (CAMEL) [16] y Open Service Architecture (OSA) [17]. El uso de la arquitectura OSA permite al operador hacer uso de los servicios de control de llamadas, interacción con el usuario, estado del usuario, control de la sesión de datos, conocer las capacidades de los terminales, gestión de la tarificación, etc.

- **MRF. Multimedia Resource Functions.**

Proporciona funciones relacionadas con el media, tales como la manipulación de de media y la activación de tonos y alertas. Cada MRF está dividido en un **MRF Controller** y un **MRF Processor,** el primero es un nodo del plano de señalización que interpreta la información entrante de los AS y S-CSCF para el control del MRFP, el cual es usado para mezclar fuentes o flujos de media, pudiéndose además emplear para gestionar el acceso a recursos compartidos.

- **BGFC y CS Gateway. Border Gateway Control Function**

BGCF es un proxy SIP que procesa peticiones para el enrutamiento desde un S-CSCF cuando éste ha determinado que la sesión no puede ser enrutada usando DNS. Su funcionalidad de enrutado esta basada en números telefónicos.

Los CS Gateways proporcionan señalización e interoperabilidad del media con redes de circuitos conmutados. Hay tres tipos:

SGW (Signaling Gateway**)** sirve de interfaz con el plano de señalización de los CS. Transforma los protocolos de bajo nivel como Stream Control Transmission Protocol (SCTP) [18] a Message Transfer Part (MTP) [19], para pasar Integrated Services Digital Network User Part (ISUP) [20] desde MGCF a la red CS.

MGCF (Multimedia Gateway Control Function) traduce el protocolo SIP a ISUP y controla el MGW usando un interfaz H.248 [21].

MGW (Multimedia Resource Function Processor) sirve de interfaz con el plano de media de la red CS para convertir las tramas RTP a Pulse Code Modulation (PCM) [22]. Puede llevar a cabo trascodificaciones cuando los codecs no son compatibles.

2.4 SIP/SDP

2.4.1 Protocolo de Iniciación de Sesión (SIP)

SIP [23] es un protocolo de nivel de aplicación desarrollado para el establecimiento, modificación y finalización de sesiones multimedia sobre una red IP. SIP no es un protocolo que proporcione de forma integrada comunicaciones multimedia, y por ello se utiliza en conjunción con otros protocolos. En el caso de Video Share se emplea junto con Session Description Protocol (SDP) [ver sección 2.4.2] para la descripción de sesiones multimedia y RTP/RTCP [ver sección 2.5] para la transmisión de la información multimedia.

SIP emplea un modelo cliente-servidor donde principalmente participan las siguientes entidades lógicas:

-**User Agent**: son los extremos de la sesión y actúan como clientes o servidores dependiendo de quién inicia la sesión: Agente Usuario Cliente (UAC) envía peticiones y recibe respuestas (inicia sesión) y Agente Usuario Servidor, envía respuestas y recibe peticiones.

-**Proxy**: actúan encaminando peticiones hacia UASs y encaminando respuestas hacia UACs. Aunque en muchos casos están diseñados para ser transparentes a los mensajes SIP, se pueden emplear para implementar lógica de servicio asociada al reencaminamiento de los mensajes (p.e. envío a buzón de voz).

-**Redirect:** recibe peticiones de clientes o servidores SIP, obtiene la asociación entre la dirección SIP del usuario que intenta contactar y sus direcciones de contacto actualizadas, y devuelve dichas direcciones de contacto del peticionario.

-**Registration:** Gestiona la información de localización de usuarios en un determinado dominio.

-**Back to Back User Agent:** por un lado actúa como servidor y por el otro como un cliente. Puede implementar cualquier lógica de servicio, no sólo asociada al enrutado, permitiendo total control sobre la transacción SIP, y por tanto rompiendo el principio de señalización punto a punto.

SIP toma su esquema de direccionamiento de Internet usando Uniform Resource Indentifiers (URIs) para identificar usuarios, servidores, proxies, etc.

-SIP URI sip:maquina.dominio.org
-SIPS URI sip: user@domain.org (conexión segura)
-TEL URI tel: +34917654321 (asociando número telefónicos)

Otros tipos de URIs como mailto, ftp o http pueden ser también usados en algunos casos.

La estructura de las peticiones y respuestas SIP es similar a la de las peticiones y respuestas HTTP, siendo extensible por diseño. La primera línea de la cabecera indica el tipo de petición o respuesta y las cabeceras Via, From, To, Call-ID, y CSeq están presentes en todos los mensajes, identificando de forma unívoca cada mensaje dentro de cada llamada:

- *Via:* Indica el transporte usado para el envío e identifica la ruta del request, por ello cada proxy añade una línea a este campo.

- *From:* Indica la dirección del origen de la petición.

- *To:* Indica la dirección del destinatario de la petición.

- *Call-Id:* Identificador único para cada llamada, contiene la dirección del host. Debe ser igual para todos los mensajes dentro de una transacción. En los mensajes de tipo OPTIONS, el campo Call-ID permite relacionar las peticiones y respuestas

- *Cseq*: Se inicia con un número aleatorio e identifica de forma secuencial cada petición.

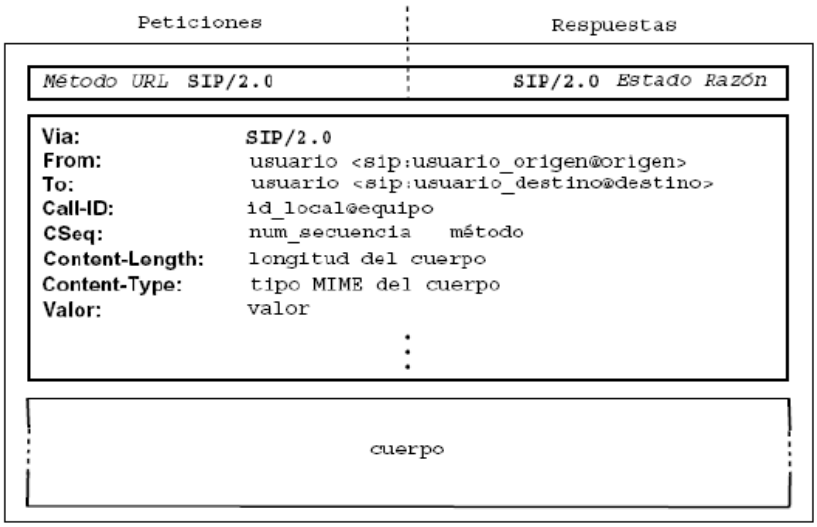


Figura 5: Cabecera de un mensaje SIP

En SIP hay cinco tipos de mensajes básicos: REGISTER, INVITE, ACK, CANCEL, BYE y OPTIONS. SIP prevé mecanismos para añadir nuevos mensajes de ampliación, así como para que los servidores que desconozcan estos nuevos mensajes puedan responder con un código de error (código 501), indicando la lista completa de mensajes admitidos en un campo, "Allow", de la cabecera de la respuesta.

- **Mensaje REGISTER**. Los mensajes REGISTER proporcionan la localización de un agente de usuario a los servidores de registro. En ellos, los agentes de usuario clientes notifican a los proxys o los servidores de redirección, la dirección o direcciones en las cuales se encuentra un usuario. En estos mensajes el campo To de la cabecera indica la dirección que se ha de registrar, mientras que el campo From indica la dirección del usuario responsable del registro.

- **Mensaje INVITE.** Los campos de cabecera de las peticiones de inicio de llamada contienen entre otros datos, la dirección de origen y destino de la llamada, el asunto, la prioridad, los equipos intermedios por los que se solicita que pase la llamada y las preferencias acerca de la localización del destinatario. El cuerpo contiene una descripción de la información multimedia que se pretende intercambiar en la sesión. Este componente, transparente para el protocolo SIP, se suele codificar siguiendo el formato SDP.

- **Mensaje ACK**. Los mensajes de este tipo sirven de confirmación para intercambios fiables de mensajes de invitación. Los clientes deben generar mensajes ACK para confirmar que se ha recibido el mensaje final de aceptación correspondiente a una invitación. El cuerpo de los mensajes ACK puede contener la descripción de la sesión si el cliente decide realizar modificaciones.

- **Mensaje BYE.** Estos mensajes indican a los servidores que un cliente desea finalizar la conexión entre dos participantes en una sesión. Se pueden generar tanto en los agentes que iniciaron la llamada como en los que recibieron la invitación.

- Mensaje CANCEL. Empleado para cancelar una llamada pendiente, aunque su recepción por parte de un UAS no garantiza que éste no responda posteriormente, sino que simplemente constituye una sugerencia realizada por el remitente para optimizar el uso de la red.

- Mensaje OPTIONS. Útil para solicitar información acerca de las posibilidades de un servidor. Los servidores de redirección y los proxys simplemente los reenvían. Otros servidores pueden responder con un mensaje en el que indiquen sus capacidades o bien con la respuesta que hubiesen dado a una invitación.

Otros mensajes comúnmente usados por SIP son: **INFO** para intercambio de cualquier información de control durante una llamada; **NOTIFY** para comunicar a UA que se ha producido un cambio de estado de la sesión; **PRACK** que es un ACK provisional para las peticiones **INVITE**; **SUBSCRIBE** Y **UNSUBSCRIBE** para realizar peticiones de suscripción en determinado sesiones de servicios.

SIP además define un código de respuestas que están asociadas a los distintos mensajes. La estructura de una respuesta es <*Sip-Version*> <*Status Code*> <*Reason-Phrase*>. En *Status Code* se indica el tipo de respuesta y en *Reason-Phrase* una explicación literal. Hay seis tipos de respuestas:

1xx - Mensajes provisionales. (100 Trying, 180 Ringing, etc.)
2xx - Respuestas de éxito. (200 Ok)
3xx - Respuestas de redirección. (302 Moved Temporarily, 305 Use Proxy, etc.)
4xx - Respuestas de fallo de método. (400 Bad Request, 401 Unauthorized, etc.)
5xx - Respuestas de fallos de servidor. (500 Sever Internal Error, (501 Not Implemented)
6xx - Respuestas de fallos globales. (600 Busy, 603 Decline, etc.)

En la siguiente figura se muestra un ejemplo básico de sesión SIP, para sesión de media:

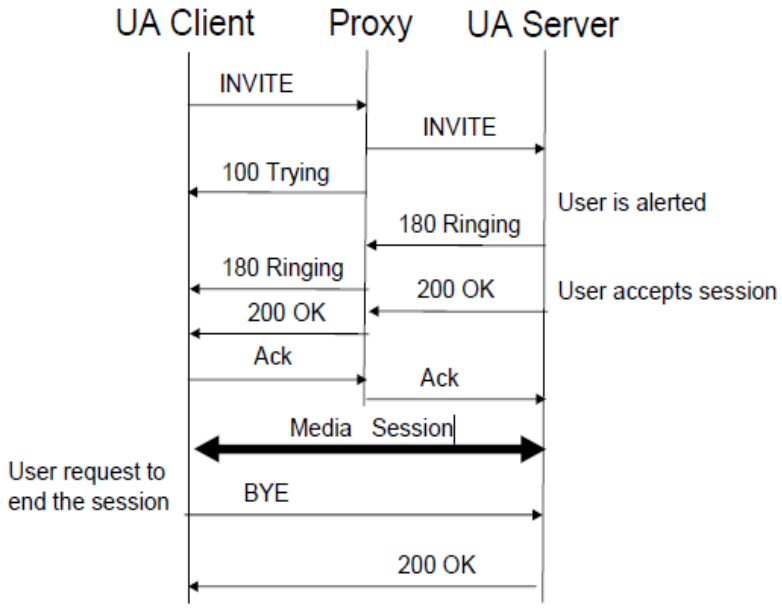


Figura 6: Ejemplo de establecimiento de sesión multimedia.

La primera transacción, anterior al establecimiento de sesión, correspondería al registro del usuario. Los usuarios deben registrarse para poder ser encontrados por otros usuarios. La siguiente transacción ya correspondiente al establecimiento de sesión comienza con una petición INVITE del usuario al proxy y finaliza con el mensaje ACK de confirmación del cliente. En este momento la llamada está establecida, pasa a funcionar un protocolo de transporte como puede ser RTP en el caso de Video Share con los parámetros (puertos, direcciones, codecs, etc.) establecidos en la negociación mediante el protocolo SDP. La última transacción corresponde a una finalización de sesión.

2.4.2 Protocolo de Descripción de Sesión (SDP)

SDP [24], es un protocolo para describir los parámetros de inicialización de los flujos multimedia. Sus dos funciones fundamentales son la de descripción de sesiones y la de negociación de capacidades.

Cada descripción SDP proporciona la siguiente información:

- Nombre y propósito de la sesión.

- Tipo y formato de los medios que la componen.

- Intervalo(s) temporal(es) de desarrollo de la sesión.

- Parámetros necesarios para poder recibir e interpretar los datos: direcciones, puertos, formatos, etc.

- Información complementaria relativa a los recursos de red necesarios para participar en la sesión: ancho de banda e información de contacto del responsable de la sesión.

Una descripción de la sesión del SDP consiste en un número de líneas del texto de la forma:

<type>=<value>

donde <type> es un carácter específico y <value> es un texto estructurado cuyo formato depende de <type>. Los campos dentro de <value> están generalmente delimitados por un espacio.

La parte de nivel de sesión comienza con una línea con “v=” y continúa con la primera sección del nivel multimedia. Cada sección multimedia comienza con una línea “m=” y continúa con la siguiente sección o con la descripción entera de la sesión. Algunas líneas en cada descripción son obligatorias y algunas son opcionales, pero todas deben aparecer exactamente en el orden abajo mostrado. Las descripciones opcionales están marcados con “*”.

```
Descripción de la sesión
   v=  (Versión del protocolo)
   o=  (Origen e identificador de sesión)
   s=  (Nombre de sesión)
   i=* (Información de la sesión)
   u=* (URI de descripción)
   e=* (Correo electrónico)
   p=* (Número telefónico)
   c=* (Información de conexión)
   b=* (Cero o más líneas con información de ancho de banda)
   z=* (Ajustes de zona horaria)
   k=* (Clave de cifrado)
   a=* (Cero o más líneas de atributos de sesión)
   Cero o más descripciones de medios

Descripción de tiempo
   t=  (Tiempo durante el cual la sesión estará activa)
   r=* (Cero o más veces de repetición)

Descripción de medios, si está presente
   m=  (Nombre de medio y dirección de transporte)
   i=* (Título)
   c=* (Información de conexión)
   b=* (Cero o más líneas con información de ancho de banda)
   k=* (Clave de cifrado)
   a=* (Cero o más líneas de atributos de sesión)
```

Figura 7: Información de descripción SDP

Los mensajes SDP se encuentran en el cuerpo de los mensajes de SIP. En la siguiente figura se muestra un ejemplo de SDP integrado en un mensaje INVITE:

```
 INVITE sip:bob@biloxi.com SIP/2.0
 Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
 Max-Forwards: 70
 To: Bob <sip:bob@biloxi.com>
 From: Alice <sip:alice@atlanta.com>;tag=1928301774
 Call-ID: a84b4c76e66710@pc33.atlanta.com
 CSeq: 314159 INVITE
 Contact: <sip:alice@pc33.atlanta.com>
 Content-Type: application/sdp
 Content-Length: 142

 v=0
 o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com
 s=-
 t=0 0
 c=IN IP4 pc33.atlanta.com
 m=audio 3456 RTP/AVP 0 1 3 99
 a=rtpmap:0 PCMU/8000
```

Figura 8: Ejemplo de descripción SDP.

La especificación original de SDP [24] no contempla la función de negociación de capacidades, pero la capacidad de negociar y seleccionar las características de cada sesión multimedia es una función básica de cualquier arquitectura de control, por lo que se requiere una solución específica para este problema. Ejemplos de intercambio de oferta y respuesta de SDP se recogen en el documento [25].

Aquí se expone un ejemplo sencillo de negociación SDP durante el establecimiento SIP de una sesión de audio y video entre dos usuarios:

```
[Offer]

    v=0
    o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
    s=
    c=IN IP4 host.atlanta.example.com
    t=0 0
    m=audio 49170 RTP/AVP 0 8 97
    a=rtpmap:0 PCMU/8000
    a=rtpmap:8 PCMA/8000
    a=rtpmap:97 iLBC/8000
    m=video 51372 RTP/AVP 31 32
    a=rtpmap:31 H261/90000
    a=rtpmap:32 MPV/90000

[Answer]

    v=0
    o=bob 2808844564 2808844564 IN IP4 host.biloxi.example.com
    s=
    c=IN IP4 host.biloxi.example.com
    t=0 0
    m=audio 49174 RTP/AVP 0
    a=rtpmap:0 PCMU/8000
    m=video 49170 RTP/AVP 32
    a=rtpmap:32 MPV/90000
```

Figura 9: Ejemplo de negociación SDP.

En este caso el originante de la sesión “alice”, quiere establecer una sesión multimedia de audio y video, indicando al receptor “bob” los códec que es capaz de soportar o que está dispuesto a emplear, para voz se trata de los codecs PCMU y PCMA mientras que para video son H261 y MPV. El receptor “bob”, atendiendo a la oferta de “alice”, selecciona los medias que es capaz de soportar su dispositivo y se lo hace saber al originante en una respuesta contenedora de un nuevo SDP. En este caso indica que para el intercambio mulitmedia se emplee los códecs PCMU y MPV.

Las variantes de la negociación pueden ser múltiples, pudiendo haber varias ofertas y respuestas y entrando en negociación diversos parámetros. La negociación permite añadir nuevos streams durante una sesión ya inicializada o el reanudamiento de sesiones en distintos puertos. Por lo general los SDP ofertados suelen ir en el cuerpo de

mensajes INVITE, y las respuestas de oferta en el cuerpo de mensajes de respuesta SIP 200 Ok.

2.5 RTP/RTCP

2.5.1 Protocolo de Transporte en Tiempo Real (RTP).

RTP [26] es un protocolo utilizado para la transmisión en tiempo real de datos multimedia encapsulados generalmente dentro de paquetes User Data Protocol (UDP) [27]. Proporciona funciones de transporte de extremo a extremo, los nodos intermedios no interpretan RTP. Conceptualmente se puede interpretar como un servicio de transporte, pero en cuanto a su implementación, se suele integrar como parte del procesamiento a nivel de aplicación de usuario. Por su condición de tiempo real, prioriza el recibir la información en el momento adecuado, no garantizando entrega fiable ni ordenada. Tampoco gestiona calidad de servicio (QoS) pero se apoya en el protocolo RTCP [ver sección 2.5.2], utilizando éste para enviar periódicamente información de control asociada con el flujo de datos, de forma que el emisor pueda ajustar su transmisión.

Una sesión RTP es una asociación entre un conjunto de participantes que se comunican con RTP. Son sesiones ligeras, es decir, no hay control estricto ni centralizado sobre quién está en la sesión, por lo que se ajusta bien al modelo multicast. RTP no indica cómo establecer las sesiones, ni determina los parámetros que las definen, de esto se encargan otros protocolos como SIP. Las sesiones RTP sólo necesitan parámetros acerca del destino de los paquetes: direcciones IP de los participantes, puerto destino RTP (suele ser un número par, por defecto 5004) y puerto destino RTCP (suele ser el puerto RTP asociado+1).

Los profiles RTP son los contextos en los que se define de forma más detallada como usar RTP. Hay profiles según los casos de uso o la codificación del media y suelen definir:

-Número de payload, que tienen significado distintos en cada perfil.

-Cómo se empaquetan la información multimedia para cada tipo de payload (cómo calcular las marcas de tiempo, cuantos datos meter por paquete, etc.).

-Incluso definen como refinar el comportamiento de RTP/RTCP en ese perfil: cambiando el intervalo de generación de paquetes RTCP, añadiendo nuevos tipos de paquetes RTCP o definiendo extensiones para RTP.

En las transmisiones RTP pueden participar un tipo de dispositivo intermedio denominado **Relay RTP**, que procesa información y la reenvía, actuando como fuente y destino. Hay dos tipos:

-**Mezcladores**: reciben paquetes de una o más fuentes y los reenvían a uno o más destinos. Puede combinar los datos recibidos, cambiarlos de formato y tienen capacidad para sincronizar flujos (audio y voz de video).
-**Traductores:** sólo convierten formatos, recibe un paquete de entrada y genera uno o más paquetes de salida. Adapta la calidad original de los flujos según las capacidades del receptor.

A continuación se describe los campos de la cabecera de un paquete RTP:

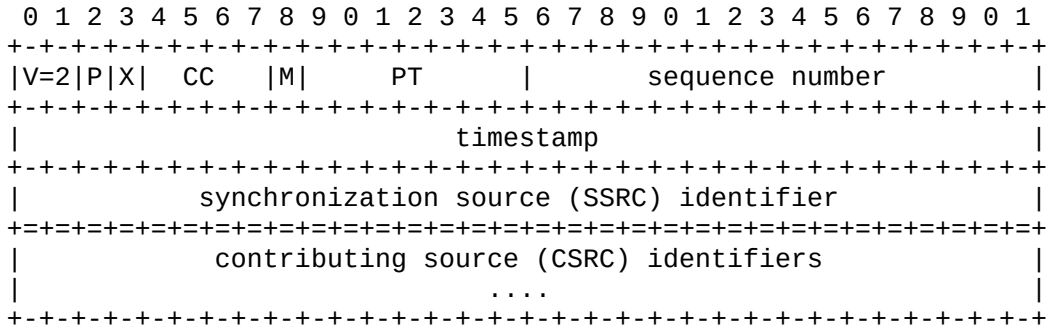


Figura 10: Cabecera de paquete RTP

- V (Versión, 2 bits): Actualmente la 2.

- P (Bit de relleno, 1 bit): Indica si el paquete lleva relleno. Algunos algoritmos de cifrado trabajan con bloques de tamaño múltiplo de un número de bits dado. Si P=1, el último byte del relleno indica cuantos octetos hay que ignorar.

- X (Extensión, 1 bit): Si X=1, hay una cabecera de extensión al final.

- CC (Cuenta de contribuyentes, 4 bits): Los mezcladores indican de cuántas fuentes se están abasteciendo.

- M (Bit marcador, 1 bit): Funcionalidad dependiente del tipo de datos transportado. Puede marcar el comienzo de una ráfaga, instante ideal para ajustar el playout delay dinámicamente (al jitter de la red, compensar diferencias de los relojes en emisor-receptor, etc.).

- PT (Tipo de datos, 7 bits): Indica el tipo de contenido (audio, vídeo) y su codificación (códec empleado, cifrado, etc.). Los tipos de datos están definidos en el documento [38].

-Número de secuencia, (16 bits): Cada fuente comienza con un número seleccionado aleatoriamente. Permite detectar pérdidas.

- Marca de tiempo (32 bits): Permite sincronización y cálculo del jitter del paquete, resulta útil para el funcionamiento de RTCP. Diferentes formatos de marca de tiempo dependientes del perfil y el payload. No debe requerir sincronización entre relojes, sólo indica valores de tiempo relativos entre paquetes, emplea un reloj virtual de muestreo, no el del sistema.

- SSRC (Synchronization Source Identifier, 32 bits): Generado aleatoriamente por la fuente, es el identificador único de una sesión RTP. Permite mantener la temporización de reproducción de cada fuente.

- CSRC (Contributing Source Identifier, 32 bits): Insertados por los mezcladores, si no los hay, no aparece y CC=0. Identifica a cada una de las fuentes originales que contribuyen al contenido del paquete (hasta 15), pudiendo haber varios por paquete.

2.5.2 Protocolo de Control de RTP (RTCP)

RTCP [28] implica la transmisión periódica de paquetes de control a todos los participantes en una sesión RTP. Sus principales funciones son:

- **Monitorización de la congestión y de QoS**: Se envían informes sobre los datos recibidos a todos los participantes en una sesión, pudiendo detectar congestión o problemas de red, de forma que se puedan tomar medidas correctivas al respecto como puede ser la adaptación de la velocidad de envío.

-Identificación: Identificador de la fuente mediante una cadena de caracteres. Puede utilizarse para asociar flujos de diferentes sesiones. También permite enviar el nombre del usuario y otros parámetros textuales de tipo informativo.

- Control de sesión mínimo: Abandono de sesión. Conocer el número de participantes.

Hay cinco tipos de mensajes RTCP:

-Sender Report (SR): Distribuye las estadísticas de emisión y de recepción de los emisores activos (los emisores también pueden ser receptores).

-Receiver Report (RR): Distribuye las estadísticas de recepción de los participantes que no sean emisores activos.

-Source Description (SDES): Proporciona información para describir la fuente. Por ejemplo, mediante CNAME, el nombre, el número de teléfono, la localización o la versión de las herramientas de aplicación.

-Bye: Es opcional e indica que un equipo abandona la sesión, pudiendo incluir la razón de por qué lo hace
.
-Específico de aplicación: Es opcional y permite añadir funcionalidades sin necesidad de definir nuevos paquetes en el protocolo.

Las estadísticas de emisión y recepción vienen recogidas dentro de los paquetes RTCP SR y RR, en bloques denominados informes, que contienen campos con diversa información para el control. La información más relevante que contienen es la siguiente:

Informe de emisor:

-Referencia de reloj, dada la marca de tiempo provisionada por Network Time Protocol (NTP) [29].
- El mismo instante temporal que la marca NTP pero utilizando el reloj empleado en RTP para las marcas. (Utilizando ambas referencias se puede realizar correspondencia intra-media o inter-media).
-Número de paquetes transmitidos.
-Total de bytes de carga útil transmitidos desde que comenzó la transmisión.

Informe de receptor. Proporciona, para cada una de las fuentes de sincronización, los siguientes datos:

-Fracción de paquetes perdidos desde que se envió el anterior informe.
-Número acumulado de paquetes perdidos desde el comienzo de la recepción.
-Número mayor de secuencia extendida que se haya recibido.
-Dispersión temporal entre las recepciones (jitter).
-Última marca de tiempo.
-Retardo desde el último informe.

Los paquetes SR proporcionan un informe de emisor y varios de receptor, mientras que los paquetes RR solo contiene informes de receptor.

2.6 Descripción técnica de Video Share en RCS-e.

2.6.1 Sesión de Video-Share

En el apartado 2 dedicado a la descripción del servicio se listaron los pasos fundamentales en los que se descompone una sesión de Video Share [7]:

Figura 11: Pasos de una sesión de Video Share

Aquí, con más detalle, se expone el funcionamiento de los protocolos participantes en cada uno de los pasos y se presentan los aspectos técnicos más relevantes.

2.6.2 Direccionamiento

Sobre la base de integración de servicios en torno a la agenda telefónica, RCS-e determina que se ha de emplear los números de teléfonos de la agenda para la

formación de las URI´s identificativas de los contactos, en las peticiones y respuestas SIP implicadas en servicios RCS-e, lo cual incluye las sesiones de Video Share [5].
Los dos tipos de URI soportados por clientes RCS-e son:

-TEL URI´s: en formato internacional (tel:+1234678980), o en formato local (tel:0234678901;phone-context=<phonecontextvalue>).

-SIP URI´s: con un parámetro "user=phone", el número de contacto será proporcionado en la parte usuario de la URI (sip:+1234678980@operator.com;user=phone o sip:0234678901;phone-context=<phonecontextvalue>@operator.com;user=phone).

El dispositivo generador de petición debe ser capaz de formar las direcciones de contacto a partir de los números telefónicos guardados en su agenda. El dispositivo receptor debe ser capaz de extraer los números telefónicos de las direcciones incluidas en las peticiones recibidas, de forma que puedan ser confrontadas con los números de su agenda de contactos.

2.6.3 Identificación de servicio.

En RCS se emplea una serie de etiquetas identificativas de cada tipo de servicio para indicar que el terminal es capaz de soportar ese tipo de servicio. En el caso de Video Share la etiqueta tiene la siguiente estructura: ***+g.3gpp.cs-voice*** [8]**.**

En el caso concreto de Video Share esta etiqueta debe incluirse en los siguientes mensajes SIP empleados durante una sesión:

-REGISTER: en la cabecera *Contact* / 200 OK: en la cabecera *Contact*.
-INVITE: en las cabeceras *Contact* y *Accept-Contact* / 200 OK: en la cabecera *Contact.*
-OPTIONS: en la cabecera *Accept-Contact* y opcionalmente en la cabecera *Contact*. /
200 OK: en la cabecera *Contact.*

2.6.4 Registro de usuarios.

El registro de usuarios se realiza cuando se activa la funcionalidad RCS-e en el terminal, siendo un registro común para todos los servicios RCS. No se necesita registrar al usuario por cada servicio RCS que soporta, esto se puede indicar mediante las etiquetas identificativas de servicio en la cabecera *Contact* del mensaje REGISTER.

En el proceso de registro RCS-e, el cliente manda un mensaje SIP REGISTER a la red, incluyendo la dirección IP y puerto del proxy SIP (P-CSCF) de entrada a la red, así como la SIP URI y/o TEL URI con la que se desea registrar. La red debería autenticar el mensaje utilizando GIBA (GPRS-IMS Bundled Authentication) [30], si este no es soportado o falla, entonces se levanta el mecanismo de autenticación DIGEST [39], basado en un reto que la red manda al cliente, al cual éste responde usando el par de configuración username/password.

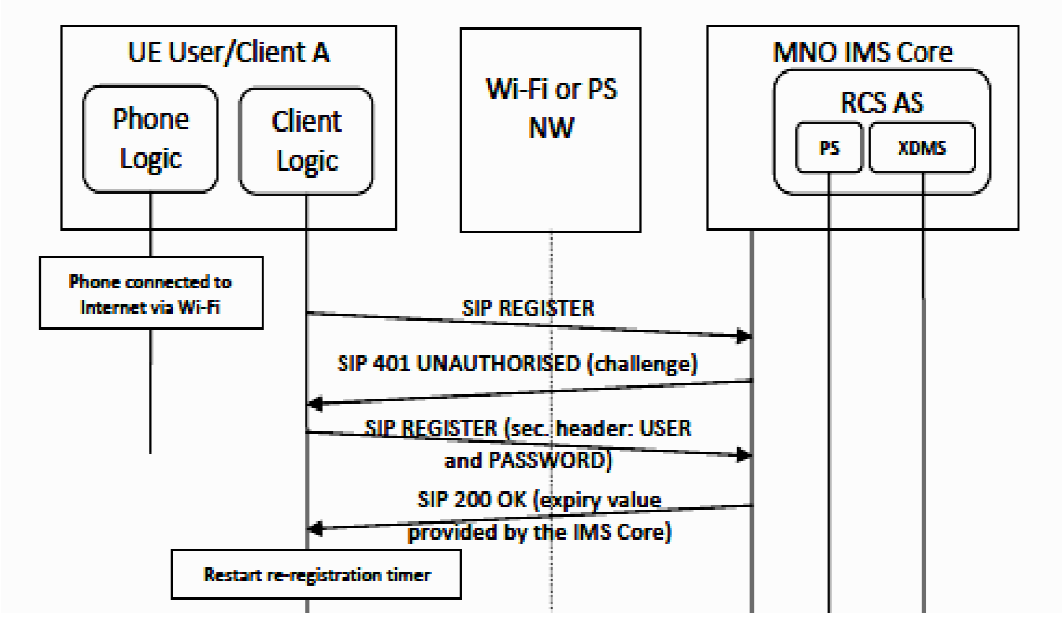


Figura 12: Procedimiento de registro del cliente RCS-e

Como parte del proceso de registro, la red proporciona un periodo de validez para el registro (SIP expire time). Si el cliente sigue registrado después de terminar el tiempo de validez, debe volver a registrarse de nuevo.

2.6.5 Petición de capacidades

Una de las características fundamentales definidas en RCS-e es la visualización (mediante iconos o indicadores activados) en la agenda del teléfono de las capacidades de nuestros contactos en cuanto al modo en el que nos podemos comunicar con ellos en un determinado momento. No se trata sólo de los servicios RCS soportados por los terminales sino de los que el usuario selecciona o los que tiene a disposición en ese momento.

Para ello RCS-e define un mecanismo de intercambio de capacidades basado en mensajes SIP OPTIONS [2]. El cliente RCS da a conocer sus capacidades al resto de clientes que aparecen en su agenda de contactos mandándoles a todos ellos un mensaje SIP OPTIONS, que incluye en su cabecera *Accept-Contact* etiquetas identificativas de los servicios que tiene disponible en ese momento. En las respuestas 200 Ok los otros clientes del mismo modo, en la cabecera *Contact*, indican sus servicios disponibles, permitiendo así actualizar las capacidades de los contactos en la agenda.

Aunque si bien es cierto que el usuario podrá enviar OPTIONS de capacidades de forma individualizada a uno o varios contactos cuando lo desee. Los clientes RCS-e deben de tener automatizado este mecanismo con el propósito de que las capacidades en la agenda aparezcan actualizadas en tiempo real. Los escenarios son los siguientes:

-Se mandara un OPTIONS a todos los contactos inmediatamente después de que el cliente RCS-e se registre por primera vez o se re-registre.

-Cuando se añada o modifique alguno de los contactos de la agenda, se les mandara OPTIONS a estos contactos nuevos o modificados.

-Mediante un mecanismo de POLLING se mandara cada cierto tiempo mensajes OPTIONS a todos aquellos contactos que no hayan sido actualizados recientemente.

-Cuando se active una nueva capacidad/servicio en el cliente o se actualice su estado, éste inmediatamente enviara un OPTION al contacto o contactos implicados.

En el caso de servicio Video Share, la petición de capacidades se realiza bajo el último caso visto. Cuando un cliente ha establecido una llamada de voz con uno de sus contactos, si el cliente RCS-e tiene la capacidad de servicio Video Share, ésta se activará e inmediatamente enviara un mensaje OPTIONS al contacto con el que ha establecido la llamada. El cliente esperara a recibir el mensaje 200 Ok que le permita saber si puede establecer una sesión de Video Share con ese contacto o si por el contrario no es posible.

El identificador de capacidad Video Share ***+g.3gpp.cs-voice*** sólo se incluirá en los mensajes OPTIONS en el caso visto, debido a la restricción de llamada de voz pre-establecida que lleva implícito el servicio.

Ambos extremos de la llamada pueden enviar OPTIONS para indicar si tienen capacidad Video Share de forma paralela, pues el envío se realiza de forma automática tras el establecimiento de la llamada independientemente de quien la inicio:

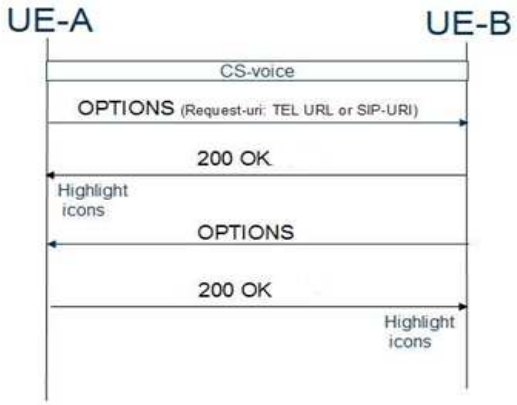


Figura 13: Petición de capacidades mediante mensajes SIP OPTIONS

2.6.6. Establecimiento de sesión. Invitación.

Después del intercambio de capacidades el siguiente paso es el levantamiento de la sesión de Video Share. En RCS-e la activación de la sesión Video Share sigue el modo IETF, contemplando también la posibilidad de establecimiento con el modo IMS.

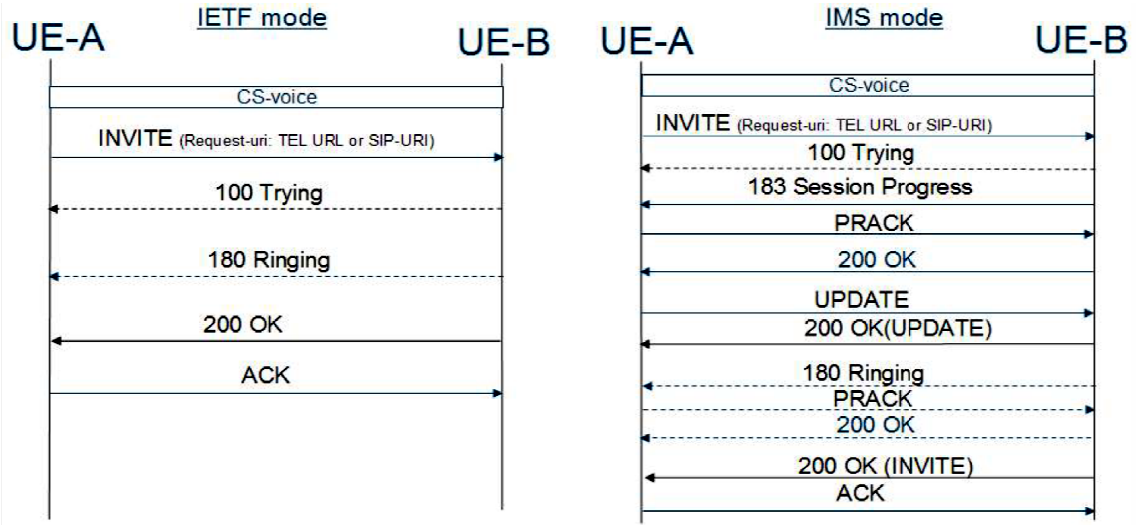


Figura 14: Establecimiento de sesión

Para inicializar la sesión una de las partes debe enviar un mensaje SIP INVITE a la SIP URI o TEL URI del otro extremo, el cual debería contestar con un 200 OK en el caso que acepte la sesión [ver sección 2.4.1]. Estos mensajes, además de la etiqueta identificativa de servicio en sus cabeceras, contienen un SDP con información sobre el media que se empleara en la transmisión del video [ver sección 2.4.2]:

INVITE

(Accept-Contact: +g.3gpp.cs-voice; explicit)
m=video *portUE-A* RTP/AVP 98 96
a=sendonly
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42C00D; packetization-mode=0
a=rtpmap:96 H263-2000/90000
a=framesize:96 176-144
a=framerate:8
a=fmtp:96 profile=0; level=45

200 OK

(Contact: +g.3gpp.cs-voice)
m=video *portUE-B* RTP/AVP 98 96
a=recvonly
b=AS:800
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42C00D; packetization-mode=0
a=rtpmap:96 H263-2000/90000
a=framesize:96 176-144
a=framerate:8
a=fmtp:96 profile=0; level=45

Figura 15: Información contenida en mensajes INVITE Y 200 OK

La línea “m=” informa acera del tipo de media a transmitir, en el caso de Video Share se trata de video, a continuación se indica el puerto que el cliente RCS pondrá a disposición para la conexión RTP y finalmente se indica los perfiles RTP que es capaz de utilizar, estos viene dado por los códec soportados. El resto de las líneas “a=” son atributos del media e indican el papel que juegan en la sesión, como receptores o emisores, los códec soportados, así como otros parámetros del video.

Por lo general el establecimiento de los parámetros de la sesión RTP para la transmisión de video se lleva a cabo mediante negociación SDP:

- El usuario A informa a B, acerca de sus posibilidades o condiciones a la hora de transmitir video, mediante el SDP del mensaje INVITE.

-El usuario B teniendo en cuenta estas condiciones, elige el codec (codificador-decodicador) y los parámetros que es capaz de soportar y manda estos, junto con el puerto donde permanecerá a la escucha, en el SDP del mensaje 200 Ok.

-El usuario A conociendo los parámetros de transmisión así como el puerto destino, establecerá la sesión RTP/RTCP con el usuario B.

-Si el establecimiento de la sesión no es aceptada, el usuario B responderá al mensaje INVITE con un mensaje de error que puede ser 486 o 603.

2.6.7 Transmisión de vídeo

Después de que la sesión de Video Share haya sido levantada, la transmisión del video entre clientes comienza, utilizándose RTP para el transporte y el protocolo de control RTCP para la supervisión de la entrega de datos, mediante envío de paquetes RTCP RR (Informe del receptor) y RTCP SR (Informe receptor) entre los clientes [ver sección 6].

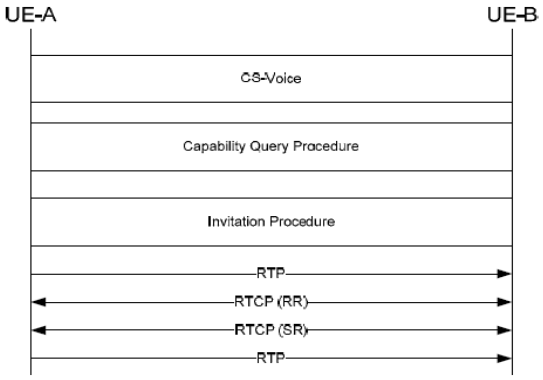


Figura 16: Transmisión de Video

Un factor determinante a la hora de la interoperabilidad del servicio Video Share es el codificado del video, dos clientes podrán establecer una sesión siempre y cuando soporten el mismo códec. Por ello que RCS-e especifique, para asegurar la interoperabilidad entre sus clientes, que al menos se deba soportar los siguientes codecs de video [31] [32]:

H.264/MPEG-4 Part 10 // AVC (Advanced Video Coding)
H.264 Profile: Baseline Profile (BP)
H.264 Level: 1b

Además recomienda soportar otros tipos de codecs de forma adicional, tal como H.263 [33], que permita diferentes niveles de calidad, usándolos de una manera adaptativa en

función del estado del terminal y las condiciones de cobertura. En el caso de que los clientes dispongan de varios formatos, la elección del formato de transmisión se realiza por negociación SDP.

La implementación y paquetización RTP se realizara de acuerdo a los codecs soportados, siguiendo los "profiles" recogidos en los correspondientes documentos para cada caso [34] [35] [36].

2.6.8 Finalización de sesión.

Cuando cualquiera de las dos partes decida detener la sesión de Video Share, la transmisión de video terminará con el envío de un paquete RTCP BYE y la sesión SIP se parara usando un mensaje SIP BYE. Después de completar estos procesos la llamada de voz aún debe existir.

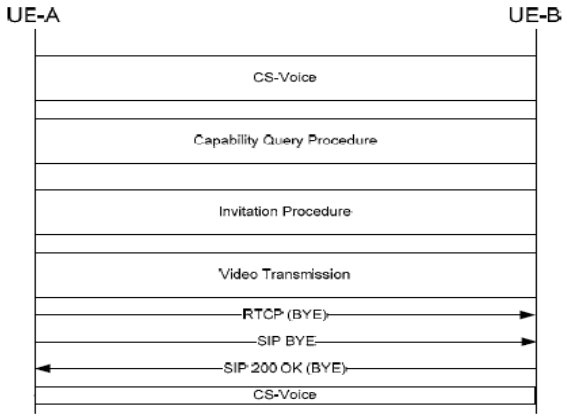


Figura 17: Finalización de sesión.

El mensaje SIP BYE debe ser enviado también siempre que por cualquier motivo la sesión de Video Share se caiga (ej. falta de cobertura 3G). Notar que los usuarios deberían poder compartir video, parar video y reiniciar la compartición de video, todo ello en una única llamada CS. La reiniciación debería incluir todo el proceso de establecimiento, incluido el SIP INVITE y el SIP BYE.

2.7 Funcionalidad de Video Share en RCS-e.

2.7.1 Tipos de sesión Video Share

RCS-e especifica tres tipos de sesiones Video Share según la fuente desde donde es originado el video [2]:

-Cámara frontal del móvil (“yo”).

-Cámara trasera del móvil (“lo que yo veo”).

-Un fichero (“video streaming”).

Por otro lado, RCS-e también define el caso de uso “bidireccional”, aunque si bien es cierto que el servicio Video Share es unidireccional, un cliente RCS-e debería ser capaz de establecer simultáneamente sesiones de video share en ambas direcciones. Una vez que el usuario A esta compartiendo video con el usuario B, el usuario B podría además iniciar una compartición de video con A, simultáneamente. En este caso cada sesión de video share es independiente y debería ser manejada de forma separada.

2.7.2 Descripción de caso de uso con clientes RCS-e.

A continuación se describe como sería un caso de uso genérico del servicio Video Share entre dos clientes RCS-e.

1. El usuario A busca en su agenda de contactos y selecciona al contacto B, mostrándosele todas las formas de contacto (capacidades). Elige llamada telefónica y establece una sesión de voz con el usuario B.

Usuario A ***Usuario A*** ***Usuario B***

Figura 18: Interfaz cliente RCS-e. Establecimiento de llamada de voz

-Cuando la llamada es establecida, inmediatamente se producirá el intercambio de capacidades entre los usuarios (SIP OPTIONS).

-Si ambos soportan el servicio Video Share, aparecerá en su interfaz un icono o indicador que así lo representa, en la imagen el icono a la izquierda de la foto del contacto (el de la derecha representa el servicio de compartición de imágenes, no tratado en este estudio).

-Si algunos de los dos usuarios no soportan Video Share, el icono de video no aparecerá en pantalla, esto puede ser debido a las siguientes causas: no tiene activado el servicio; no tiene cámara o no está habilitada; el usuario no está emplazado bajo cobertura 3G.

-Notar que algunas de las condiciones anteriores se pueden dar de forma temporal. Si durante la llamada de voz se produce un cambio en éstas, RCS-e mediante su mecanismo automatizado de intercambio de capacidades, basado en polling y notificación de eventos, lo pondrá en conocimiento de los terminales cambiándose el estado del icono.

2. Una vez establecida la llamada y con el servicio habilitado, cualquiera de las partes puede iniciar una sesión. En este caso el usuario A pulsa sobre el icono de video para iniciar la sesión, acto seguido se despliega un menú con los diferentes tipos de sesión. Si seleccionase "Media gallery" se desplegaría un explorador de búsqueda del fichero de video. Una vez seleccionado el tipo de sesión, el mensaje SIP INVITE de establecimiento de sesión se manda al usuario B, éste al recibirlo le aparecerá en pantalla un menú de aceptación.

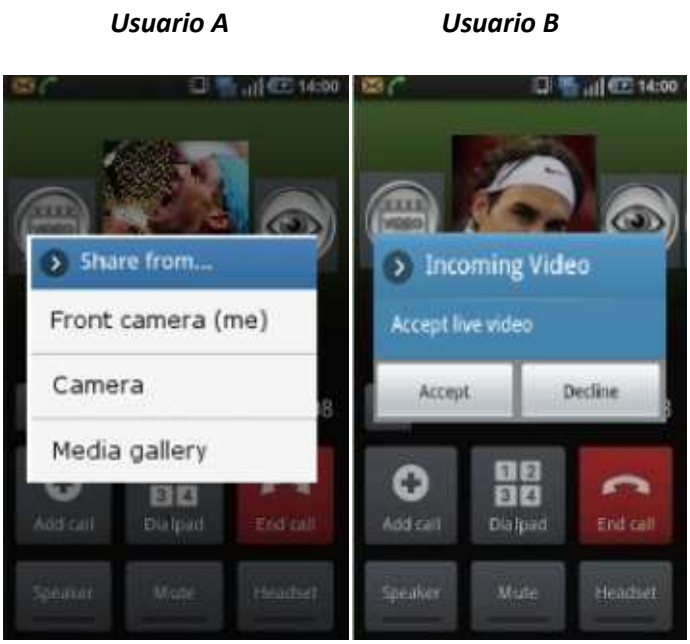


Figura 19: Interfaz cliente RCS-e. Iniciación de sesión de video share

-Si el usuario B declina la invitación, un mensaje SIP de error (603 decline) se enviaría al usuario A, notificándosele con un mensaje por pantalla al usuario. Inmediatamente se cerraría la sesión con SIP BYE.

-Si por el contrario B acepta, entonces mandaría un 200 OK al usuario A, que al recibirlo iniciaría la transmisión.

-Notar que puede suceder que si llega el 200 OK y la negociación de códecs no es fructífera, es decir, los terminales no soportan códecs en común, se mandará el mensaje SIP de error correspondiente y se cerrara la sesión con SIP BYE.

3. La trasmisión de video queda establecida entre ambos usuarios, pudiendo ambos visualizar en vivo la captura de video.

Usuario A ***Usuario B***

Figura 20: Interfaz cliente RCS-e. Transmisión de video share

-Cualquiera de los dos usuarios puede parar la sesión desde su lado pulsando sobre el icono “stop” habilitado, cerrándose la sesión inmediatamente mediante el envío de SIP BYE. La llamada de voz seguirá establecida.

-Si por el contrario la llamada de voz se cae/termina o la capacidad de Video Share en alguno de los extremos se pierde, se mandará un mensaje SIP de error, notificándose a los usuarios con un mensaje por pantalla de inhabilitación de servicio. La sesión de video se cierra, pero la llamada de voz continua en el caso de que su terminación no sea la causa.

2.7.3 Interacción con otros servicios.

- **Llamada de voz a 3**: [8] Video Share es sólo soportado sobre llamadas entre dos personas. Si un usuario pone la llamada en espera o acepta otra llamada entrante durante la compartición de vídeo, la llamada de voz se mantendrá pero la sesión Video Share caerá.

-Interacción con llamada en espera: Cuando haya otra llamada de voz entrante ya sea para el emisor o para el receptor de Video Share, el usuario debería tener la capacidad de

aceptar la llamada. Si el usuario acepta la segunda llamada, la sesión de Video Share debería terminar pero ambas llamadas de voz quedarían establecidas.

-**Interacción con llamada múltiple:** Cuando se produce una llamada múltiple, la opción de Video Share debe presentarse inhabilitada. Si se dieran INVITES entrantes se contestaría con mensajes de error 486 o 603.
-**Transición de UMTS a GSM:** Cuando alguno de los dispositivos pasa de UMTS o EDGE/DTM a GSM, la sesión de Video Share se cae.

-**Video Share cuando esta fuera de Cobertura UMTS:** La capacidad de Video Share se deshabilita cuando el dispositivo esta fuera de cobertura. La opción de iniciación o recepción de Video Share no debería aparecer cuando se encuentra en GSM.

-**Señalización de llamada en espera:** Después de un intercambio satisfactorio de OPTIONS, si el terminal recibe un "Facility message" de que la otra parte a puesto la llamada en espera, entonces el cliente RCS-e debería ser capaz de indicar que el Video Share no está habilitado en ese momento. Una vez el terminal reciba un nuevo "Facility message" de reanudamiento de llamada, el cliente restaurara la funcionalidad Video Share.

2.8 Evolución de Video Share. Fase 2

2.8.1 Escenario del servicio en su fase 2.

El servicio Video Share, hasta ahora visto, se fundamentaba en transmisión de video punto a punto entre usuarios, no siendo necesario entidades adicionales a las ya empleadas en el establecimiento de sesión (SIP). En la fase dos [9], la transmisión de video se apoya en la arquitectura IMS [ver sección 2.3], empleando servidores de aplicaciones de video share VS-AS y entidades para la gestión de media como son MRFC y MRFP, vistos en el apartado 4. Además de las entidades IMS otros dos bloques Web-Portal y VS-Storage, descritos más abajo, son añadidos de forma complementaria. Este nuevo escenario implica una gran ampliación de las posibilidades del servicio y un mayor número de casos de uso.

Web-Portal: proporciona un interfaz Web (Wb1) a los suscriptores, para vía login, acceder y manejar los video-clips subidos a la unidad de almacenamiento Video-Storage. También mediante este interfaz facilita a clientes web el acceso a la compartición de flujo de video con terminales móviles, mediante conexión Real Time Streaming Protocol (RTSP) [37] facilitada por MRFP. Toda la información acerca de permisos de usuarios y del contenido accesible, se lo proporciona VS-AS.

Video-Storage: mantiene espacios de almacenamiento para cada usuario suscripto a VS fase 2, a los que puede subir sus videos. También facilita el almacenamiento desde clientes Web vía RTSP.

Sin entrar en más detalles y remitiéndonos al apartado 4 del presente estudio se presenta la arquitectura de Video-Share fase 2:

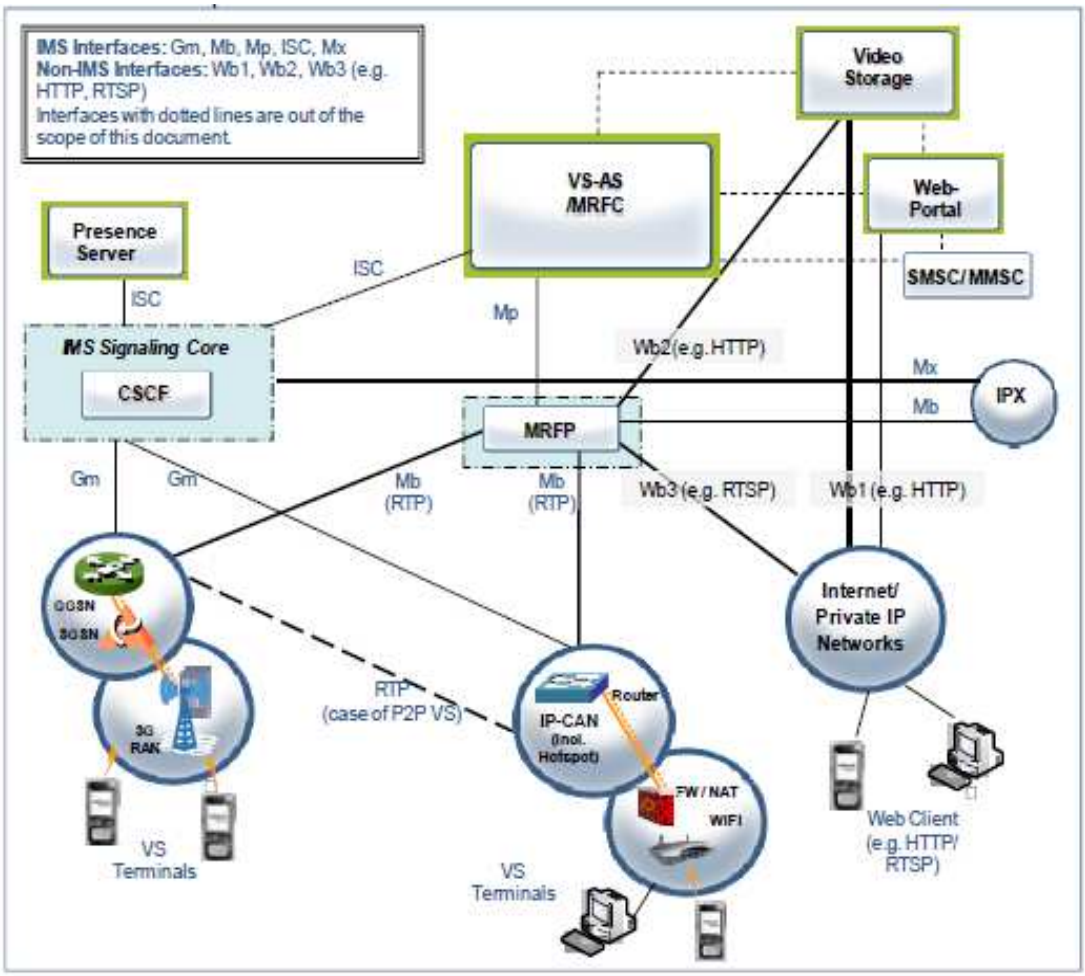


Figura 21: Arquitectura de Video Share fase 2

2.8.2 Intercambio de capacidades basado en OMA Presence.

Otro elemento importante añadido en la arquitectura de Video Share es el servidor de presencia, que permitirá el intercambio de capacidades vía OMA-presence [6] [9]. El entorno de presencia facilita la petición de capacidades del siguiente modo:

- Un dispositivo de VS fase 2 toma el papel de Source de presencia y publica su información de presencia en PIDF (Presence Information Data Format). En el PIDF, la capacidad Video Share debe ser indicada por la inclusión de <service-description> con <service-id> puesta a *"org.gsma.videoshare"* y <version> puesta a 2.0.

-El servidor de presencia recibe y almacena la información de presencia PIDF y sirve esta información ante las peticiones que reciba.

- Los dispositivos VS a su vez juegan también el papel de Watcher, para obtener la información de presencia proporcionada por el servidor de presencia, con el objetivo de obtener las capacidades del usuario remoto.

Para poder implementar los pasos vistos anteriormente, los dispositivos tienen que ampliar su funcionalidad SIP, soportando los métodos SUBSCRIBE, NOTIFY y PUBLISH.

El envío de PUBLISH para publicar información acerca de las capacidades de Video Share se hará en los siguientes casos:

-Tras un registro IMS satisfactorio, indicando sus capacidades.

-Cuando el servicio deje de funcionar o se deshabilite.

-Cuando el funcionamiento del servicio se habilite o se reanude.

2.8.3 Nuevos tipos de sesión Video Share.

-Sesión punto a punto.

El VS AS juega el papel de B2BUA durante el levantamiento de las sesiones SIP entre terminales, de forma que durante el intercambio SIP le llegan mensajes y los reenvía. Esto conlleva consigo un mayor control sobre la sesión, permitiendo la inclusión de calidad de servicio QoS en la sesión y el manejo del ancho de banda entre otros.

VS fase 2 incluye casos de uso donde no es necesario el establecimiento previo de llamada, y donde el audio es incorporado al video de forma sincronizada, transportándose del mismo modo, mediante intercambio de paquetes PS. El audio se transmite por RTP como otro media distinto, por lo que es necesario también incluir la información al respecto en los SDP, pudiéndose dar negociación del mismo.

-Sesión punto a multipunto

Los terminales que soporten VS fase 2 pueden establecer sesiones de uno a muchos, no estando combinadas con llamadas de voz CS. Los mecanismos de división y gestión de la transmisión de video son llevadas a cabo por MRFP y MRFC. En cuanto a SIP, los terminales deben incluir en sus mensajes INVITE listas de URI´s para indicar los participantes en la conferencia.

Es posible añadir y borrar participantes de la conferencia, si se trata del originante de la misma, o borrarse si solo sé es receptor. Para poder añadir y borrar participantes el dispositivo utiliza mensajes SIP REFER.

-Sesión con video originado desde Video Storage

Es una variante de la sesión punto a punto, donde el terminal originante de la transmisión es sustituido por un video clip pregrabado y almacenado en Video Storage, esto debe ser indicado en los mensajes INVITE con el atributo *X-vrsrc-id* en el SDP, que identifique el video-clip.

Capítulo 3

Diseño del sistema

En este capítulo se verá el proceso seguido para el diseño del sistema RCS-e Video Share Gateway. En primer lugar se determina cual ha de ser la funcionalidad del sistema, luego se recogen los requisitos necesarios para que el sistema pueda ofrecer dicha funcionalidad, y por último de acuerdo a lo anterior se presenta la arquitectura modular del Gateway.

En cuanto a la funcionalidad del sistema, veremos que papel juega el sistema a implementar y el escenario de funcionamiento donde será integrado, posteriormente en sucesivos subapartados se detallan los casos de uso que debería cubrir, y finalmente se expone como será ofrecida dicha funcionalidad. Los requisitos del sistema serán enumerados en un segundo apartado, donde se distinguirán entre requisitos funcionales que definen el comportamiento interno del sistema y requisitos no funcionales que aportan las cualidades que debiera tener el mismo. El capítulo terminará con la presentación de la arquitectura del sistema, donde se describirán cada uno de los módulos que la conforman, cómo interactúan entre si y cuales son sus funciones.

3.1 Funcionalidad del sistema.

El objetivo del sistema a implementar es el de dar soporte de servicio Video Share a aplicaciones web externas al entorno RCS-e, de forma que se puedan dar sesiones de compartición de video entre éstas y usuarios propiamente RCS-e a través del core IMS/RCS.

El sistema funcionará como gateway entre la aplicación y los usuarios RCS-e. Del lado de la aplicación el Gateway se comunica a través de un API, cuyas funciones representan acciones y notificaciones del servicio VS a alto nivel. Esta comunicación que incluye intercambio de datos entre aplicación y Gateway se fundamenta en protocolos y estándares Web. Con respecto al core y usuario RCS-e, el Gateway empleará SIP para señalización y RTP para la transmisión y recepción de video. Los mecanismos empleados en este caso son los descritos por RCS-e y la especificación IR.74 para clientes VS [capitulo 2]. De esta forma, el sistema Gateway permite la convergencia de SIP/RTP y HTTP [40] para llevar el servicio VS propio de las telecomunicaciones al mundo de las aplicaciones Web.

Más allá de soporte SIP y RTP el sistema ofrece a cada una de las aplicaciones adscritas al mismo un conjunto de funcionalidades equiparables al que un cliente RCS-e ofrece al usuario del servicio VS:

-Envío de video pregrabado.

-Envío de video en vivo.

-Recepción de video, con posibilidad de ser visualizado en tiempo real y/o grabado en un fichero.

-Registro en el core IMS/RCS.

-Gestión de la lista de contactos y control de las capacidades de servicio de los mismos.

Las aplicaciones accederán a estas funcionalidades de forma remota a través del API del Gateway. Las acciones y notificaciones recogidas en este API son funcionalmente similares a las acciones y notificaciones que la interfaz gráfica de los clientes RCS-e, embebidos en terminales móviles, ofrecen al usuario final [ver sección 2.8.2].

Con el objetivo de adaptar la funcionalidad del servicio VS a los requerimientos de las aplicaciones web y aligerarlas en lo posible en cuanto a carga se refiere. El Gateway se apoyará en dos bloques adicionales que complementan el soporte RTP del sistema para la transmisión y recepción de video.

- **Repositorio de contenidos.**

Será empleado para el envío de video pregrabado o el almacenamiento del video recibido, durante una sesión VS. El administrador del Gateway asignara un espacio de almacenamiento del repositorio a cada una de las aplicaciones que se den de alta en el sistema y les proporcionara acceso al mismo:

-Las aplicaciones cuando solicite al Gateway envío de video pregrabado no tendrán que proporcionarle de forma directa el fichero a enviar, simplemente indicarán la URL donde el fichero ésta almacenado en el repositorio, siendo el Gateway quien se ocupa de obtenerlo del mismo.

-En el caso de recepción la aplicación no requerirá realizar las tareas de grabado y almacenamiento sino que indicará al Gateway el identificador del espacio de almacenamiento y el nombre del fichero con el que desea guardarlo y éste se ocupa de todo el procesamiento, proporcionando en última instancia a la aplicación la URL donde se encuentra almacenado.

El repositorio empleado ha sido desarrollado por la empresa Solaiemes [41] y esta implementado sobre Apache Jackrabbit [42], empleando la API JCR (Java Content Repository) [43]. El repositorio cuenta con una consola para la gestión de espacios de almacenamiento y contenidos, y proporciona un API de acceso, sobre peticiones HTTP POST y HTTP GET [40], para aplicaciones.

- **Servidor de video en vivo. "Liveserve".**

Este módulo, integrado en el Gateway, le proporcionará la capacidad de envío y recepción de video en *streaming* hacia/desde las aplicaciones web durante un sesión VS, para sesiones de video en vivo. Los protocolos empleados son:

-Real Time Streaming Protocol (RTSP) [37], es un protocolo que permite a un cliente controlar de forma remota un servidor de streaming, mediante comandos tales como "play" y "pause", permitiendo el acceso a los archivos del servidor bajo demanda.

- Real Time Messaging Protocol (RTMP) [44] es un protocolo propietario desarrollado por Adobe Systems que se utiliza para transmitir audio y vídeo a través de Internet al cliente de Adobe Flash Player, de uso muy extendido en aplicaciones web.

Liveserve, es un servidor de video desarrollado por la empresa Solaiemes que permite la convergencia entre flujos de video RTP, RTSP y RTMP. El funcionamiento es el siguiente:

- Recepción de video en vivo. Liveserve actúara como servidor, emitiendo flujos RTSP o RTMP hacia la aplicaciones que actuán como clientes, utilizando como fuente el flujo RTP de la sesión VS entrante.

- Emisión de video en vivo. Liveserve como cliente accederá al flujo RTSP/RTMP proporcionado por las aplicaciones web (mediante servidores RTSP/RTMP o camára IP), y convierte éstos a flujos RTP, que es el protocolo empleado para servicio videoshare.

A continuación se presenta el escenario en el cual el Gateway despliega toda su funcionalidad, donde aparecen todas las entidades con los que interactúa y los protocolos de comunicación que emplea. Después en sucesivos subapartados veremos en detalle la funcionalidad del Gateway a través de los distintos casos de uso del servicio VS.

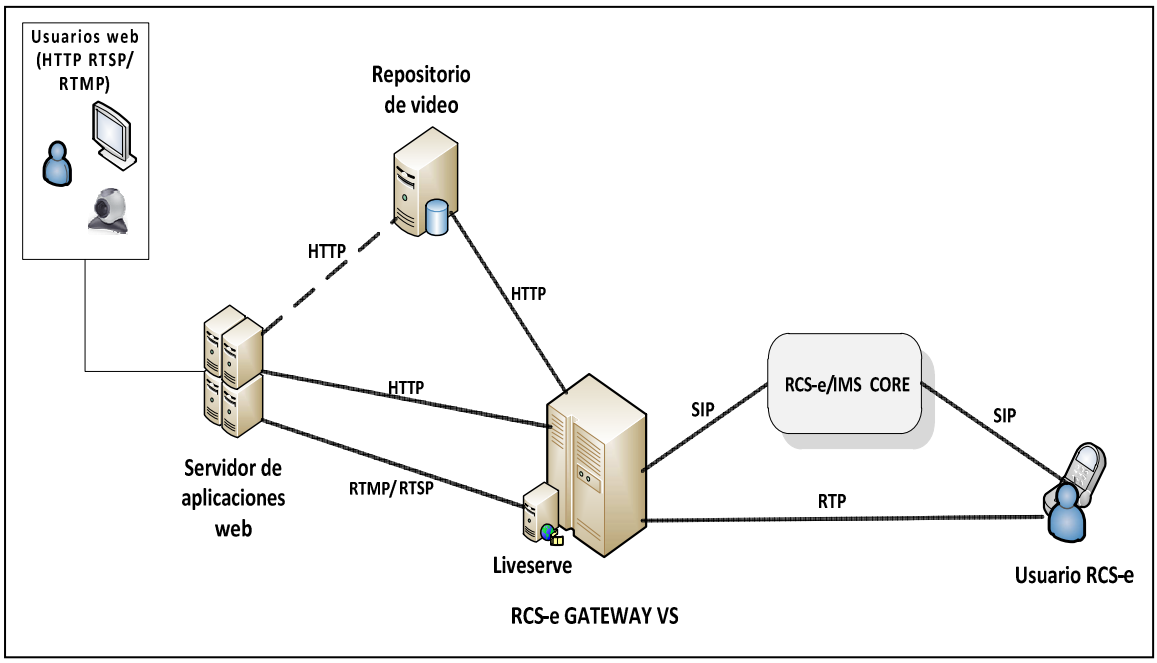


Figura 22. Escenario de funcionamiento del RCS-e VS Gateway.

3.1.1 Envío de video almacenado.

La aplicación requiere establecer una sesión video-share con uno de sus usuarios para transmitirle video contenido en un fichero, el cual previamente fue almacenado en el repositorio de contenidos.

1. La aplicación invoca la acción del API correspondiente al envío de fichero de video. Los parámetros son la URL donde está localizado el fichero de video y la SIP-URI del usuario destinatario.

2. El Gateway emplea la URL para obtener del repositorio el fichero que posteriormente será empleado como fuente de la transmisión.

3. El Gateway lleva a cabo el establecimiento de sesión SIP, incluyendo SDPs en los mensajes SIP para el negociado de medias y puertos de transmisión.

4. Si el establecimiento es satisfactorio manda una notificación de inicio de transmisión a la aplicación y comienza a transmitir el video mediante RTP.

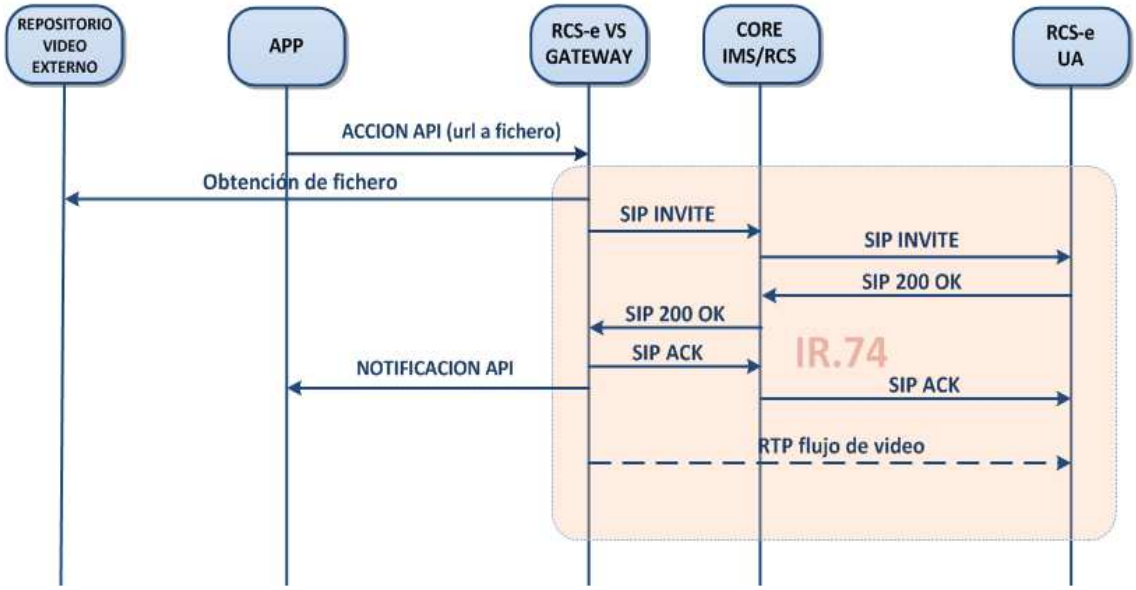


Figura 23. Envío de video almacenado

3.1.2 Envío de flujo de video en tiempo real.

La aplicación quiere establecer una sesión VS para transmitir video a uno de sus usuarios. En este caso la fuente de transmisión es un flujo de video en tiempo real proporcionado por la misma aplicación, esto permitirá el caso de uso en el que se quiera transmitir video en vivo capturado por una cámara. La captura del video y la publicación del mismo, son parte de la aplicación, que actuará como servidor RTSP o RTMP, mientras que el sistema tomará el papel de cliente.

1. La aplicación invoca la acción correspondiente al envío de flujo de video, indicando por parámetros la URL de tipo RTSP o RTMP, desde donde será publicado el video que desea transmitir y la SIP-URI del usuario destinatario.

2. El Gateway realiza el establecimiento SIP de la sesión video share y el negociado de medias con el usuario destino.

3. Si el establecimiento se completa con éxito, entonces se manda una notificación para que la aplicación comience la publicación del video.

4. El Gateway inmediatamente empleando la URL de flujo de video se conecta a éste. Al mismo tiempo que le van llegando los paquetes de video desde la aplicación, los reenvía al usuario destino, basándose en la negociación previa y utilizando el protocolo de transporte RTP, propio de Video Share.

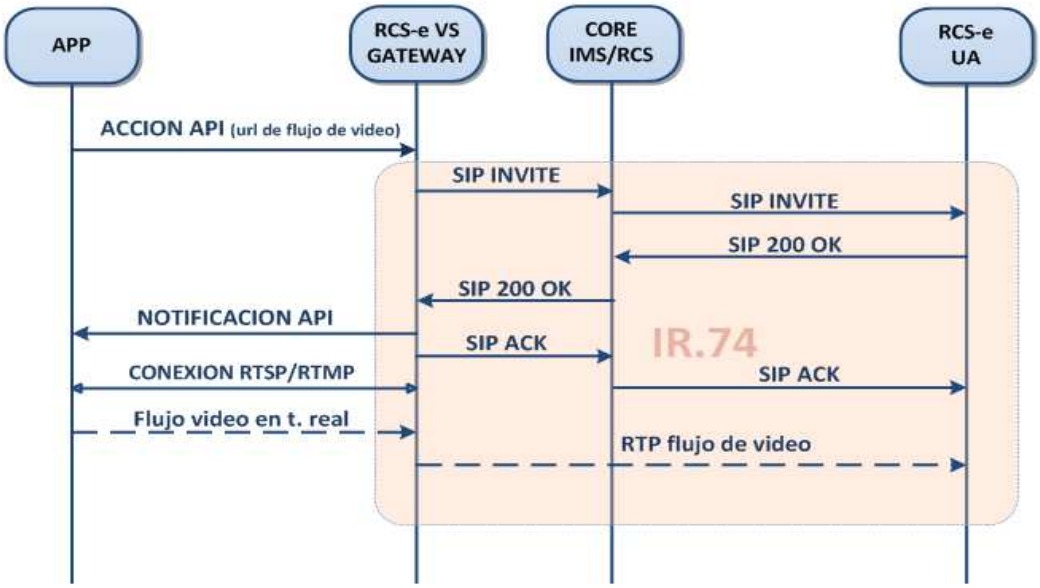


Figura 24. Envío de flujo de video en tiempo real

3.1.3 Recepción de video.

Se vuelve a iniciar un establecimiento de sesión VS, pero en este caso el originante es el usuario RCS-e (terminal móvil) de una de las aplicaciones adscritas al sistema, el cual desea transmitir video hacia la aplicación en cuestión. La aplicación en caso de aceptar la sesión podrá decidir si quiere reproducir el video en tiempo real, si quiere guardarlo en un fichero o ambas cosas a la vez.

1. El Gateway recibe del lado del core un mensaje SIP INVITE, procedente de un usuario RCS-e VS.

2. El Gateway tras comprobar a cual de las aplicaciones va dirigida la invitación, manda una notificación de sesión entrante a la aplicación involucrada indicándole el usuario originante de la misma. La aplicación en este punto decidirá si acepta o rechaza la sesión VS.

- Opción 1. Aceptación de video.

3. La aplicación invoca la acción correspondiente, indicando al Gateway que desea hacer con el video a recibir:

   -Si desea salvarlo, entonces le pasa al Gateway por parámetros el nombre del fichero con el que se guardara el video y el username identificativo del espacio de almacenamiento del repositorio donde desea hacerlo.

   -Si quiere reproducir el video en tiempo real, indicará al Gateway el protocolo a emplear para la publicación (RTSP o RTMP).

4. El Gateway completa el establecimiento de sesión SIP y el negociado de medias ya iniciado por el usuario RCS-e de la aplicación.

5. Si el establecimiento se completa con éxito, el Gateway notifica a la aplicación el comienzo de la recepción de video. Si se se activo la opción de publicación en tiempo real, el Gateway como servidor, incluye en la notificación la URL de publicación, a la cual la aplicación debe conectarse para su reproducción.

6. El Gateway comenzará a recibir el flujo de video vía RTP, sirviéndolo en tiempo real a la aplicación, si está se ha conectado previamente vía RTSP o RTMP. De forma paralela el Gateway graba en local el video en un fichero.

7. Cuando el flujo RTP termina, si en la invocación de la acción se indico almacenamiento de fichero. Este será llevado al repositorio y acto seguido se envía una notificación a la aplicación indicándole la URL donde se encuentra localizable el fichero en el repositorio.

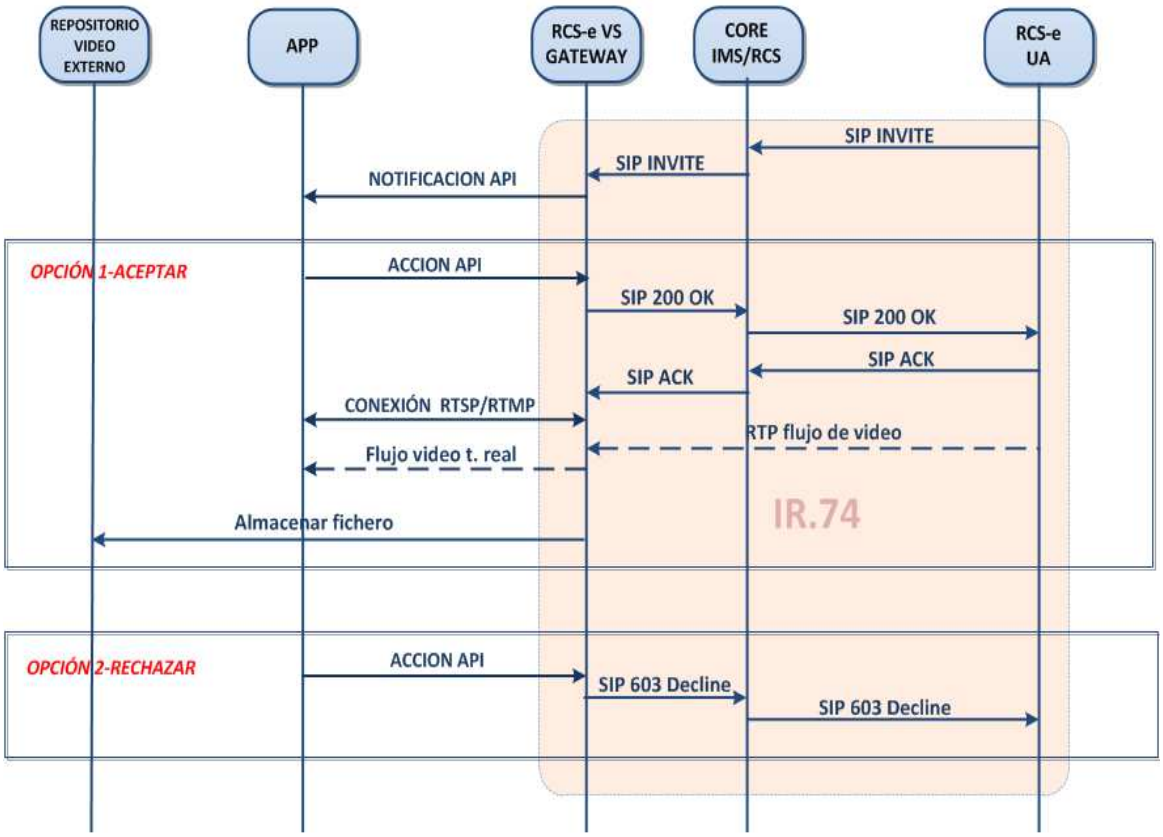


Figura 25. Recepción de video.

- Opción 2. Rechazo de sesión

3. La aplicación invoca la acción del API correspondiente. En esta acción se puede incluir opcionalmente la causa por la que no se desea establecer la sesión.

4. El Gateway mandara un mensaje de error SIP Decline al usuario originante de la sesión, incluyendo la causa.

3.1.4 Terminación de sesión

La finalización de sesión VS puede ser iniciada por cualquiera de los dos extremos de la sesión, es decir, por la aplicación o por el usuario RCS-e participante, independientemente de cuál de ellos sea el transmisor y el receptor.

-Caso 1. Iniciado por la aplicación

La aplicación podrá cerrar la sesión siempre que quiera, ya sea durante la transmisión RTP, interrumpiendo la misma, o una vez haya sido ésta finalizada, tras recibir la notificación al respecto, desde el Gateway.

1. La aplicación invoca la acción de cierre de sesión indicando por parámetro cual de las sesiones que tiene abierta quiere cerrar. Para ello se emplea un identificador de sesión previamente generado por el Gateway y pasado a la aplicación, como parámetro de notificación, cuando se produjo el establecimiento.

2. El Gateway se encarga del cierre ordenado de la sesión, iniciando éste con el envío de un mensaje SIP BYE al usuario remoto.

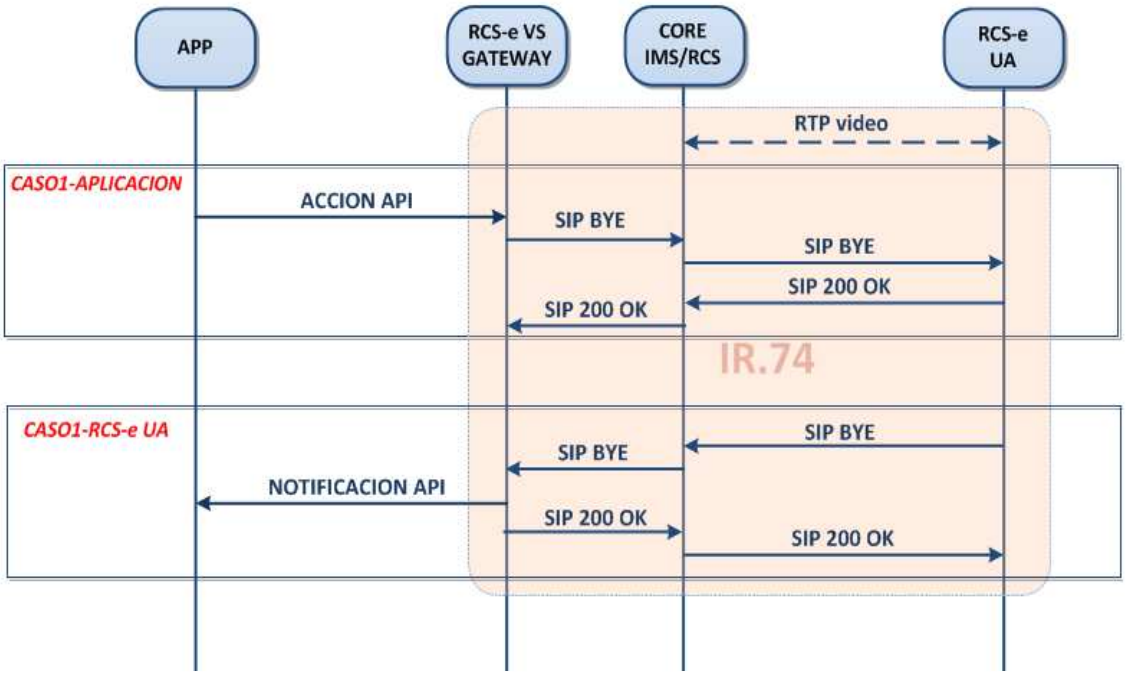


Figura 26. Terminación de sesión.

-Caso 2. Iniciado por el usuario RCS-e

1. El Gateway recibe un mensaje SIP BYE de cierre de sesión.

2. El Gateway comprueba el usuario originante y la aplicación destino del mensaje del cierre. Acto seguido le manda a la aplicación la notificación de cierre de sesión.

3. El Gateway completa la finalización SIP de la sesión.

Un tercer caso de cierre de sesión se dará siempre que se produzca un fallo durante el establecimiento de sesión. Cuando el Gateway detecta que no puede darse el establecimiento de sesión enviará el mensaje SIP de error correspondiente, lo notificará a la aplicación usuaria correspondiente y se quedará esperando a que el cierre se inicie desde el usuario. Si por el contrario, es el Gateway el que recibe el mensaje SIP de error procedente del core o el usuario remoto, entonces inicia automáticamente el cierre de sesión y luego lo notifica a la aplicación.

3.1.5 Despliegue de una aplicación.

Del lado del core IMS/RCS, una aplicación a través del Gateway se comporta como un usuario RCS-e más, y como tal, antes de poder establecer cualquier sesión requiere estar registrado en el core. Este registro puede ser interpretado como la activación o despliegue de la aplicación.

1. La aplicación cuando es activada o desplegada, hace una llamada a la acción de registro del API.

2. El Gateway, empleando la información de aprovisionamiento SIP con la que se subscribió la aplicación, inicia el registro de la misma.

3. Si el core IMS/RCS requiere autenticación, un mensaje SIP de error 401 llegará al Gateway. Entonces éste empleará el mecanismo de DIGEST [39] utilizando el username y password asociados a la aplicación y volverá a mandar un mensaje SIP REGISTER con la autenticación.

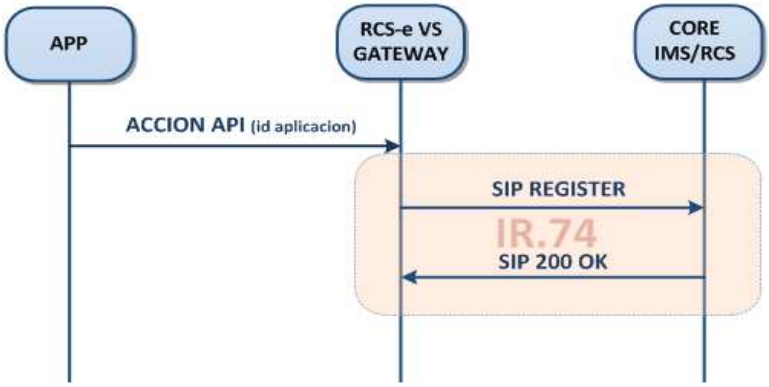


Figura 27. Registro de una aplicación

Cuando la aplicación es cerrada o desactivada el procedimiento es similar, invocándose la acción correspondiente a de-registramiento. El Gateway en este caso también mandara un mensaje SIP REGISTER, pero incluyendo de-registramiento en sus cabeceras.

El Gateway también se encarga de controlar el tiempo de expiración del registro. De modo que cuando expire, si antes la aplicación no invoco la acción de de-registro, el Gateway procede a refrescar el registro de la aplicación del mismo modo que para la primera vez.

3.1.6 Llamada de voz preestablecida.

Como se vio en el capítulo 2, dedicado al estado del arte, la especificación IR.74 que define el servicio VS determina que una sesión de VS solamente se puede dar previo establecimiento de una llamada de voz. Esta restricción, de carácter más comercial que técnico, es establecida en los clientes RCS-e embebidos en terminales móviles, no habiendo implicaciones a nivel de red, ya que llamada y sesión VS son totalmente independientes, la primera es establecida a través de la red de conmutación telefónica, mientras que la sesión VS se lleva a cabo a través del core IMS/RCS.

Aunque la especificación RCS-e a la hora de definir el servicio VS se acoge a lo estipulado por IR.74, tiene previsto incorporar cambios al respecto, a medida que se empiece a tomar como referencia la release 3 de RCS, donde la compartición de video es independiente de la llamada de voz [ver apartado 2.1.2]. Algunos de los clientes RCS-e en desarrollo, en previsión de la nueva definición del servicio, ya han empezado a eliminar esta restricción.

Para el desarrollo de este proyecto se ha seguido la premisa de independencia entre llamada de voz y sesión de video.

3.1.7 Gestión de los usuarios de las aplicaciones.

En el capítulo 2 se vio como la agenda de contactos del terminal móvil es integrada en los clientes RCS-e, con el objetivo de poder mostrar al usuario las capacidades de servicio de sus contactos, es decir, las formas en que se podía comunicar con ellos. Esto era posible por el intercambio de OPTIONS entre los clientes a nivel SIP.

En nuestro caso, el Gateway mantendrá internamente la lista de contactos de todas las aplicaciones adscritas al mismo, los contactos serán interpretados de cara a las aplicaciones como los usuarios de las mismas. De lado del core IMS/RCS, el tratamiento de estas listas por parte del Gateway será el mismo que el cliente RCS-e tiene con los contactos de la agenda del móvil. Se realizará intercambio de mensajes SIP OPTIONS con todos los contactos para el descubrimiento de la capacidad VS.

Desde las aplicaciones, mediante invocación de acciones del API, estas pueden gestionar su lista de contactos de forma remota: añadir un usuario, borrar un usuario,

obtener la lista de usuarios. En este último caso, el Gateway como respuesta a la invocación de la acción, devolverá a la aplicación el listado completo de sus contactos/usuarios junto al estado de capacidad VS de cada uno en ese momento.

En lo que respecta a la petición de la capacidad VS, tendremos en cuenta la premisa de independencia respecto al establecimiento de llamada, por lo que el Gateway se limitará a emplear un sencillo mecanismo de polling: cuando la aplicación en cuestión se registre, se mandará mensajes SIP OPTIONS a todos los usuarios de su lista, repitiéndose éste envío de forma periódica mientras se encuentre registrado. Con la inclusión de la etiqueta identificativa del servicio VS en la cabecera *Contact* de los mensajes SIP OPTION, se mantendrán informados a los usuarios respecto a la disponibilidad de la aplicación [ver apartado 2.6.5].

A su vez, el Gateway tendrá conocimiento de la capacidad VS de los usuarios de las aplicaciones, ya sea por las respuesta SIP de éstos a los mensajes OPTION (200 Ok si disponen, SIP error si no es así) o porque desde los clientes RCS-e de los terminales lleguen mensajes SIP OPTION (los usuarios tienen en su agenda como contacto alguna de la aplicaciones adscritas al Gateway). En todo caso, si el estado del usuario en cuanto a su disponibilidad cambiase, el Gateway lo actualizará en la lista y se lo comunicará a la aplicación mediante una notificación.

El Gateway cuando reciba un mensaje SIP OPTION destinado a alguna de sus aplicaciones, contestará automáticamente (no interviene la aplicación) con 200 Ok si la aplicación está registrada en ese momento y el usuario originante del mensaje forma parte de la lista de contactos de esa aplicación, en caso contrario se responderá con error. Adicionalmente el sistema ofrece la posibilidad de configurar auto-aceptación de usuarios, lo que implica que ante la llegada de un mensaje SIP OPTION desde un usuario, si este no pertenece a la lista de contactos, se añada automáticamente pasando a formar parte de la lista de usuarios, notificándolo así a la aplicación.

3.1.8 API del sistema.

En última instancia, los desarrolladores de aplicaciones (terceras partes) serán los encargados de integrar la funcionalidad ofrecida por el Gateway en la aplicaciones, según los requerimientos de las mismas. El desarrollador software planificará las llamadas al API (acciones) y gestionará las notificaciones recibidas.

En posteriores capítulos se detallarán aspectos técnicos acerca de como emplear el API, y se pondrá un ejemplo de uso a través de una sencilla aplicación. Aquí nos limitaremos a presentar el conjunto de todas las acciones y notificaciones que ofrece el sistema. En cuanto a las acciones, se resume la funcionalidad que hay tras cada una de ellas y los datos que requieren por parte de la aplicación. Respecto a las notificaciones, se describe los casos en los que estás serán enviadas a las aplicaciones y la información que les proporcionan.

ACCIONES

sendFileVideoshare	Establecimiento de sesión VS con el usuario destinatario y envío de fichero de video durante la sesión.

	Parámetros **-username**: identificador de la aplicación que quiere enviar el fichero de video. **-destUri**: tel-uri o sip-uri identificativa del usuario destinatario **-subject**: asunto de la sesión que será incluido en el INVITE. **-filename:** nombre del fichero a enviar. **-url:** url donde se localiza el fichero.
sendStreamVideoshare	Establecimiento de sesión VS con el usuario destinatario y conexión del flujo de video a la sesión VS. *Parámetros* **-username:** identificador de la aplicación que quiere enviar el fichero de video. **-destUri**: tel-uri o sip-uri identificativa del usuario RCS-e destinatario **-subject:** asunto de la sesión que será incluido en el INVITE. **-url:** url donde se localiza el fichero
closeVideoStream	Terminación de la sesión videoshare seleccionada. Una petición SIP BYE es enviada al participante remoto de la sesión *Parámetros* **-username :** identificador de la aplicación que quiere cerrar la sesión de videoshare. **-sessionId:** identificador de la sesión videoshare a cerrar.
acceptVideoshare	Aceptar la sesión de videoshare entrante. Una respuesta SIP 200 Ok es enviada al participante remoto *Parámetros* **-username:** identificador de la aplicación que quiere aceptar la invitación recibida. **-sessionId:** identificador de la sesión SIP entrante aceptada. **-publishActions:** cadena estructurada que contiene una descripción acerca de las acciones de publicación del video a recibir. **-storeActions:** cadena estructurada que contiene descripción acerca de acciones de almacenamiento del video a recibir.
rejectVideoshare	Rechazar la sesión de videoshare entrante. Una respuesta SIP 601 Decline es enviada al participante remoto. *Parámetros* **-username:** identificador de la aplicación que quiere rechazar la invitación recibida. **-sessionId:** identificador de la sesión SIP entrante rechazada. **-reason:** opcional descripción de la razón por la que es rechazada.

register	Log-in del usuario RCS-e virtual en el core IMS/RCS. Esto se puede interpretar como la operación de despliegue de la aplicación que se encuentra tras el usuario virtual. Una petición SIP REGISTER es enviada al core IMS/RCS. *Parámetros* **-username:** identificador del aplicación a registrar/desplegar.
unregister	Log-out del usuario RCS-e virtual en el core IMS/RCS. Esto se puede interpretar como la operación de repliegue de la aplicación que se encuentra tras el usuario virtual. Una petición SIP REGISTER es enviada al Core IMS/RCS. *Parámetros* **-username:** identificador de la aplicación a de-registrar/replegar.
getContactList	Obtención de la lista de contactos/usuarios actualizada. Para cada uno de los contactos se le dará a la aplicación, su sipuri, su displayname y si si esta habilitado para servicio VS. *Parámetros* **-username:** identificador de la aplicación solicitante de su lista de contactos. *Resultado* **-contactList:** lista de contactos, con información de los mismos
addContact	Añadir un usuario a la lista de contactos de la aplicación. *Parámetros* **-username:** identificador de la aplicación que quiere añadir un contacto a su lista. **-contactUri:** sipuri o teluri identificativa del contacto a añadir. **-contacDisplayName:** nombre del contacto con el que será visible en la aplicación.
removeContact	Borrar un usuario de la lista de contactos de la aplicación *Parámetros* **-username:** identificador de la aplicación que quiere borrar un contacto a su lista. **-contactUri:** sipuri o teluri identificativa del contacto a borrar

Tabla 1. Acciones del API

NOTIFICACIONES	
incomingVideoshare	Esta notificación sera enviada a la aplicación cuando una nueva petición SIP para establecer una sesión videoshare sea recibida. *Parámetros* **-username** : identificador de la aplicación a quién va dirigida la invitación. **-sessionId:** identificador de la sesión SIP entrante que que contiene la invitación **-remoteUri:** sipuri que identifica al usuario remoto que envío la invitación de videoshare.
videoshareStarted	Esta notificación será enviada cuando la sesión videoshare haya sido satisfactoriamente establecida. *Parámetros* **-username:** identificador de la aplicación partícipe de la sesión establecida. **-sessionId:** identificador de la sesión videoshare establecida **-remoteUri:** sipuri que identifica al usuario remoto partícipe de la sesión videoshare establecida. **-rtmpUrl**: url donde el flujo de videoshare es publicado en formato RTMP. Será enviado sólo en el caso de que la publicación RTMP haya sido activado cuando se invoco la aceptación de la sesión VS. **-rtspUrl:** url donde el flujo de videoshare es publicado en formato RTSP. Será enviado sólo en el caso de que la publicación RTSP haya sido activada cuando se invoco la aceptación de la sesión VS.
videoshareStopped	Esta notificación será enviada cuando la sesión videoshare haya terminado. *Parámetros* **-username:** identificador de la aplicación partícipe en la terminación de la sesión. **-sessionId:** identificador de la sesión videoshare terminada **-remoteUri:** sipuri que identifica al usuario remoto partícipe de la sesión videoshare terminada. **-rtmpUrl**: url donde el flujo de videoshare fue publicado en formato RTMP. Será enviado sólo en el caso de que haya habido publicación RTMP. - **rtspUrl** : url donde el flujo de videoshare fue publicado en formato RTSP. Será enviado sólo en el caso de que haya habido publicación RTSP.
videoshareFileStored	Esta notificación será enviada cuando un nuevo fichero es almacenado después de que la sesión de videoshare haya terminado.

	Parámetros **-username:** identificador de la aplicación partícipe. **-sessionId:** identificador de la sesión videoshare. **-remoteUri:** sipuri que identifica al usuario remoto partícipe de la sesión videoshare. **-fileUrl:** url donde se ha almacenado el fichero de video **-thumbUrl:** url donde se ha almacenado la imagen thumb del fichero de video almacenado.
videoshareError	Esta notificación será enviada cuando durante el establecimiento de sesión o durante la transmisión RTP, se produce un error.
	Parámetros **-username:** identificador de la aplicación involucrada. **-sessionId:** identificador de la sesión videoshare en la que se ha producido el error.
registerSucess	Esta notificación será enviada cuando el cliente virtual/aplicación reciba un 200 Ok ante un intento de REGISTER. Los refrescos del REGISTER también están incluidos, por lo que se mandará con la misma periodicidad que estos si no hay fallo.
	Parámetros **-username:** identificador del cliente/aplicación que ha intentado registrarse
registerError	Esta notificación será enviada cuando el cliente virtual/ aplicación reciba una inesperada respuesta de error ante un intento de REGISTER. Notar que los refrescos del REGISTER están también incluidos, lanzándose esta notificación si sucede mientras estos se dan.
	Parámetros **-username:** identificador del cliente/aplicación que ha intentado registrarse.
unregisterSuccess	Esta notificación será enviada cuando un cliente virtual/ aplicación reciba un 200 Ok ante un intento de deregistramiento.
	Parámetros **-username:** identificador del cliente/aplicación que ha intentado deregistrarse.
unregisterError	Esta notificación será enviada cuando el cliente virtual/ aplicación reciba una inesperada respuesta de error ante un

	intento de deregistramiento. *Parámetros* **-username:** identificador del cliente/aplicación que ha intentado deregistrarse.
contactAdded	Esta notificación será enviada a la aplicación cuando se haya añadido un contacto satisfactoriamente. *Parámetros* -**username**: identificador de la aplicación a quien pertenece. -**contact**: información del contacto añadido (sipuri y displayname).
contactRemoved	Esta notificación será enviada a la aplicación cuando se haya eliminado un contacto satisfactoriamente. *Parámetros* -**username**: identificador de la aplicación a quien pertenece. -**contactUri**: sipuri del contacto eliminado
contactChanged	Esta notificación será enviada a la aplicación cuando se haya añadido un contacto satisfactoriamente. *Parámetros* -**username**: identificador de la aplicación a quien pertenece. -**contact**: información del contacto (sipuri y displayname).

Tabla 2. Notificaciones del API

3.2 Requisitos del sistema

En este apartado se listan los requisitos o requerimientos funcionales que establecen el comportamiento interno del sistema, mediante el cual se implementa toda la funcionalidad anteriormente expuesta.

Los requisitos funcionales serán completados con requisitos no funcionales de carácter más cualitativo. Aunque el propósito del proyecto es desarrollar un prototipo que cubra los casos de uso de forma básica, se mencionarán algunos requisitos de calidad que se tendrán en cuenta a la hora de implementar el sistema y que serán necesarios en fases posteriores de producción y comercialización.

3.2.1 Requisitos funcionales.

1. **Administración de usuarios/aplicaciones**

Los usuarios del sistema serán las aplicaciones. Por cada usuario se creará internamente clientes virtuales RCS-e de VS como entidades lógicas que comparten los recursos del sistema.

Cada usuario/cliente virtual es representado internamente por su perfil SIP, con la información para el acceso al core y las funciones de señalización requeridas (previo aprovisionamiento del operador), y por su perfil de medias, con información acerca de codecs y parámetros a emplear en la transmisión del video.

2. **Gestión de sesiones VS.**

El sistema internamente se encarga de la gestión de todas las sesiones VS que se establezcan entre aplicaciones y los usuarios de éstas.

El sistema asociará cada sesión a la aplicación correspondiente y controlará el estado de las mismas a partir de los eventos que se produzcan en las diferentes interfaces del sistema: API del usuario, señalización SIP o transmisión RTP.

3. **Control del registro y gestión de listas de contactos.**

El sistema controlará la disponibilidad de cada aplicación a partir del estado de registro de las mismas

Por cada aplicación el sistema internamente gestionara su lista de contactos/usuarios. Se encarga de agregar y eliminar los contactos por petición de la aplicación, y controla la disponibilidad de los mismos en cuanto a su capacidad de servicio VS, a partir de los eventos producidos por el mecanismo de intercambio de mensajes SIP OPTIONS.

4. **Soporte SIP para el registro de aplicaciones, establecimiento y control de sesiones VS e intercambio de OPTIONS.**

El sistema debe realizar las funciones de Agente de Usuario SIP, tanto servidor como cliente, para cada una de las aplicaciones adscritas al sistema: generación, parseo, envío y recepción de mensajes SIP.

La base de la implementación será la RFC 3621, el formato de mensajes y los mecanismos de funcionamiento de cara a la interacción con el core y los usuarios RCS-e se hará de acuerdo a la especificación IR.74 en el entorno RCS [ver sección 2.7].

5. Soporte RTP/RTCP para la transmisión y recepción de flujo de video.

El sistema tendrá la capacidad para enviar o recibir un flujo de video RTP por cada sesión VS establecida entre aplicaciones y usuarios de las mismas

Paquetizado/depaquetizado RTP y envío/recepción de paquetes RTP sobre UDP.

Control RTCP mediante envío y recepción de paquetes de tipo SR y SS, para cada sesión VS establecida.

6. Acceso a repositorio multimedia.

El Gateway actuará como cliente del repositorio de video, solicitando a éste la obtención y el almacenamiento de ficheros para todas las aplicaciones que requieran envío y/o recepción de video pregrabado durante las sesiones VS usuario-contenidos. Para ello se integrará en el sistema el API de acceso proporcionado por el repositorio.

7. Formatos y codecs de medias.

- Transmisión RTP de video

El codec y los parámetros tomados como referencia para la transmisión RTP de video, son los especificados por RCS-e para servicio VS:

Codec. H.264
Clock rate 90000 Hz
Framerate 15/1
Resolución 176x144
Bitrate adaptativo a condiciones de red.

El profile para la transmisión RTP será: H.264 video (MPEG-4 Part10) descrito en el documento RFC 3984.

Adicionalmente para los casos donde las condiciones de cobertura 3G son peores, se proporcionará soporte para video codificado en H.263-2000. El profile para transmisión RTP viene dado por la RFC 4629.

- Video almacenado.

Como fuente de trasmisión, el sistema soportará cualquiera de los formatos y codecs de uso más extendido (formato: flv, mp4, avi, 3gp… / códec video: H.263, H263+ y H.264).

Para el grabado o almacenamiento del video recibido, el formato elegido será mp4, el códec de video será H.264. Esta elección se ha realizado en base a la calidad que ofrecen y por la extensión de su uso.

- Publicación y transmisión de video en vivo.

El módulo Liveserve [ver sección 3.1.1], empleado para la publicación y recepción de flujos de video en tiempo real a través de Internet, proporciona tanto en emisión como recepción video codificado en H263.

- Las funciones a realizar por el sistema son las siguientes:

Demultiplexado/multiplexado de medias para la extracción/grabado de video de/a ficheros, según formatos.

Trascodificado del video para adaptar los codecs empleados en cada caso de uso a los requeridos para transmisión RTP y viceversa

8. Negociado de medias y puertos para transmisión RTP de video.

El sistema proporcionará soporte SDP a cada una de las aplicaciones para el negociado de los parámetros de transmisión con sus usuarios:

Como receptor, el criterio seguido por el sistema será elegir el primero de los medias ofertados en el SDP recibido que sea soportado por la aplicación, según su perfil de medias asociado. En el caso de que la aplicación sea el originante, se incluirán en el SDP todos los medias soportados por la aplicación, según su perfil de medias, dejando la elección en manos del usuario [ver sección 2.7]

Los SDP también incluyen los puertos de la conexión RTP/RTCP, desde donde se enviará los paquetes de video, o donde la aplicación permanecerá a la escucha para recepción de los mismos. El Gateway debe gestionar eficientemente sus puertos, comprobar cuales están libres y cuales ocupados, para que puedan darse múltiples sesiones en paralelo, sin conflicto.

9. Comunicación con las aplicaciones.

El servicio proporcionado por el Gateway a las aplicaciones es de carácter asíncrono, no ajustándose al modelo cliente-servidor de peticiones y respuestas, por ello que será proporcionado mediante un API formado por acciones y notificaciones. Las acciones serán invocadas desde las aplicaciones, implementándose la lógica que hay tras ellas en el Gateway, mientras que las notificaciones serán invocadas desde el Gateway, implementándose la lógica para su gestión en las aplicaciones.

El sistema debe proporcionar un mecanismo de comunicación con las aplicaciones, mediante el cual éstas puedan acceder a las acciones del API y por el cual el Gateway pueda hacer llegar las notificaciones a las aplicaciones de forma remota, intercambiando datos a través de la red.

.

3.2.2 Requisitos no funcionales

1. Interoperabilidad.

La interoperabilidad es una de las máximas perseguidas por el proyecto RCS-e. Los distintos clientes RCS-e, embebidos en diferentes terminales y a través de los cores RCS/IMS de las distintas operadoras, deberían tener plena interoperabilidad, siempre que desarrolladores de clientes, fabricantes y operadoras de telecomunicaciones se ajusten a lo estipulado por la especificación RCS-e.

En este sentido, el Gateway debe ser capaz de interoperar a nivel SIP con cualquier core RCS/IMS, previo aprovisionamiento por parte del operador (asignación de SIPURIs o TELURIs para registro y localización de las aplicaciones) y debe ser capaz de transmitir y recibir video desde/hacia cualquier cliente RCS-e.

Del lado de las aplicaciones también se buscara máxima interoperabilidad, de forma que cualquier aplicación software, independientemente del lenguaje de programación y la plataforma de ejecución, tengan acceso a la funcionalidad del Gateway. Para ello se emplearán estándares abiertos para la comunicación y el intercambio de datos entre aplicaciones y Gateway.

2. Escalabilidad y rendimiento.

En un escenario de máxima carga, todas las aplicaciones adscritas al Gateway puede establecer sesiones VS con todos sus usuarios RCS-e, de forma paralela y en ambos sentidos (usuario-aplicación y aplicación-usuario). Teniendo en cuenta que el sistema debería dar soporte al mayor número de aplicaciones posibles y que éstas pueden ser de uso masivo (gran número de usuarios), la escalabilidad del sistema será un factor determinante a la hora de poder dar uso comercial a la plataforma. Aunque la escalabilidad del sistema en la fase que aborda este proyecto es tomada en un segundo plano con respecto a la funcionalidad, consideraremos el uso de tecnologías y herramientas clusterizables que permitan abordar la escalabilidad del sistema en fases posteriores.

Video Share es un servicio en tiempo real siendo la latencia un factor clave, por lo que se requerirá un alto rendimiento del sistema que limite los tiempos de ejecución. Se tendrán en cuenta estos aspectos a la hora de diseñar el software y se empleará una plataforma de ejecución que proporcione un alto rendimiento.

3. Modularidad.

El proyecto RCS cuenta con una serie de releases [ver apartado 2.1.2] donde se definen como ha de ser la evolución de sus servicios, la nuevas funcionalidades que incorporaran éstos conforme se vayan asentando las bases del proyecto. En este sentido, tendremos en cuenta la evolución prevista para el servicio VS, a la cual el Gateway debería poder adaptarse añadiendo nuevas funcionalidades y cambios en la ya implementadas, sin que supongan grandes modificaciones en la estructura general del

sistema. Por otro lado, el sistema tratándose de un prototipo debe estar abierto a los requerimientos de las aplicaciones conforme éstas vayan adoptando la funcionalidad del Gateway.

En base a lo anterior se hace necesaria una arquitectura para nuestro sistema que cuente con una alta modularidad. El sistema se descompondrá en una serie de módulos y componentes con funcionalidad y dependencias bien definidas de forma que se evite la propagación de los cambios y se favorezca la reutilización de los módulos.

3.3 Arquitectura general del sistema.

En este apartado se describe la arquitectura del sistema, diseñada de acuerdo a la funcionalidad que debe aportar y los requisitos que debe cumplir.

Para el diseño del RCS-e Gateway VS se ha llevado a cabo una separación entre el plano de control y señalización y el plano de datos. La arquitectura del sistema estará conformada por una serie de módulos software implementados en lenguaje de programación Java sobre dos servidores:

-Servidor de video, se ocupa en exclusiva de las funciones asociadas al plano de datos, en este caso se trata de la gestión, transmisión y recepción de flujos de video.

-Servidor de control y señalización, se encarga de la señalización SIP/SDP asociada al servicio VS, controla el acceso al servicio por parte de las aplicaciones y se ocupa de todas las funciones que complementan el servicio, tales como el registro, la gestión de usuarios o el acceso a repositorio.

El sistema además cuenta con una base de datos gestionada desde el servidor de control y señalización, donde se aloja toda la estructura de datos que empleará el Gateway.

A continuación se presenta el diagrama completo de la arquitectura del sistema, cuya descripción se realizará en los siguientes apartados, donde se detallan los módulos que componen ambos servidores, la funcionalidad de cada uno de estos módulos y las dependencias entre ellos. Antes de esto, se describe el modelo de datos sobre el que actuarán estos módulos.

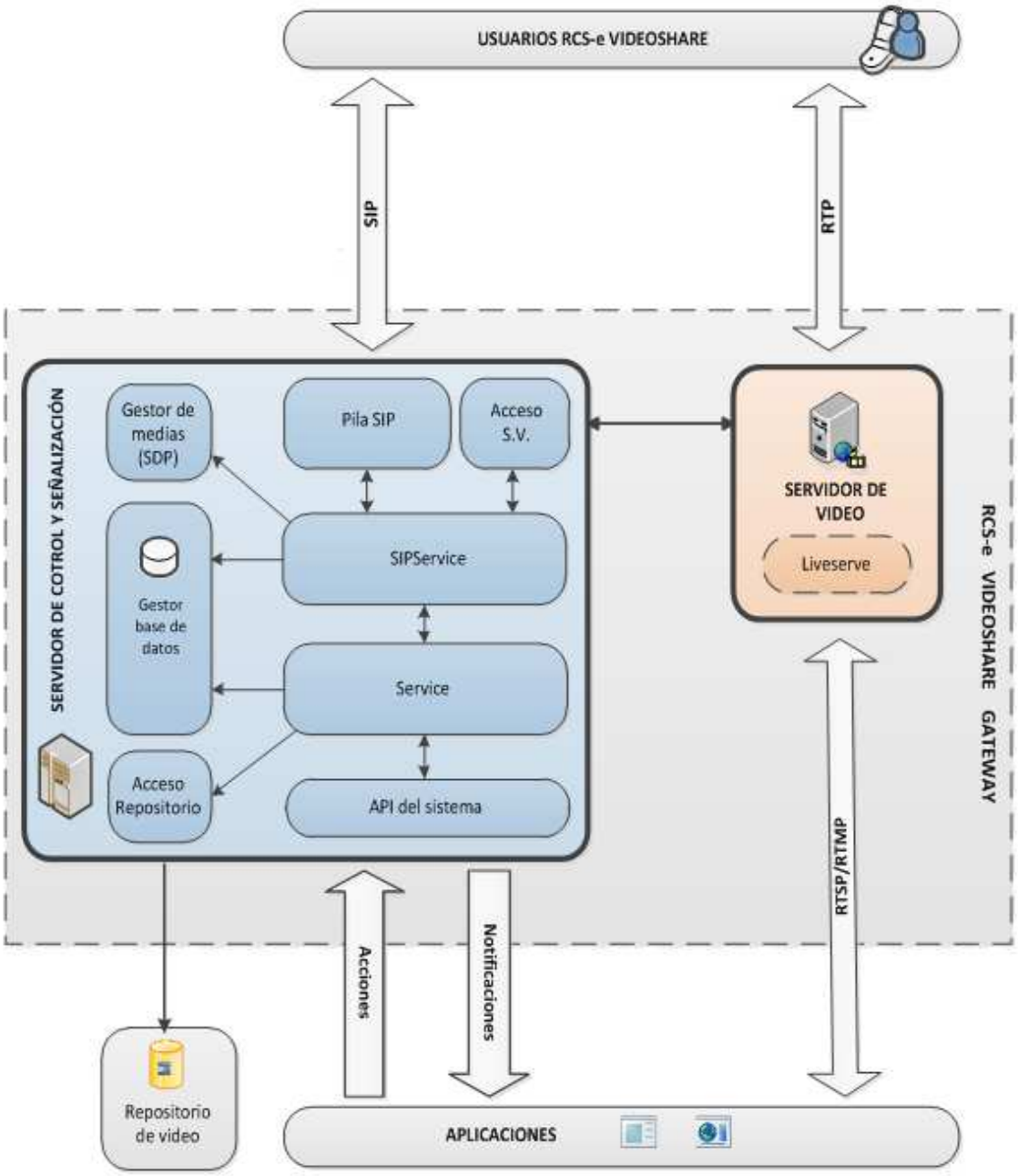


Figura 28. Arquitectura general del RCS-e VS Gateway

3.3.1 Modelo de datos

- **Datos de las aplicaciones usuarias del sistema.**

En base de datos, el administrador del Gateway creara un registro por cada aplicación que sea dada de alta en el sistema. Este registro contiene la información que el sistema empleará a la hora de prestar el servicio a dicha aplicación: perfil SIP del usuario, perfil de medias del usuario y urls de la aplicación para recibir notificaciones del servicio.

El perfil SIP, contiene entre otros los siguientes parámetros:

-La SIPURI y TELURI, con la que se identifica en el core y a través de la que se encuentra localizable para sus usuarios.

-La dirección IP y el puerto del proxy de acceso al core, que empleará para establecer la sesiones VS con sus usuarios.

-El password a emplear en el registro, cuando el core requiera autenticación.

-Los periodos de expiración, tanto para el refresco de registros, como para el envío periódico de OPTIONS a sus contactos.

El perfil de medias contiene un campo donde se guarda el mensaje SDP que se empleará como plantilla, configurado éste a partir de los codecs soportados por la aplicación. Además el perfil contiene el valor de los parámetros que se utilizarán en la transmisión del media y que se contemplarán en la negociación durante el establecimiento:

-Bitrate máximo de transmisión de video.

-Resolución de video soportado, especificando el tamaño y el rate de los frames de video.

-Payloads a emplear en el paquetizado RTP.

- **Información de sesiones.**

El servidor de control y señalización implementa un sistema de mapeo objeto-relacional sobre la base de datos, creando objetos persistentes con información acerca de las sesiones y procesos que se llevan a cabo durante la prestación del servicio, para cada una de las aplicaciones, permitiendo así el control y la concurrencia de operaciones.

-Información de registro: por cada aplicación adscrita al sistema habrá una entrada en base de datos con información acerca de su estado de registro en el core: identificador de la aplicación en el sistema (username), estado de registro (registrado, no registrado, fallo de registro), fecha de ultimo registro e identificador de sesión de registro.

-Información de option: por cada envío o recepción de un mensaje OPTION se creará una entrada, que será eliminada cuando el proceso de actualización de capacidades termine: identificador de la aplicación adscrita al sistema (username), SIPURI del contacto (usuario de la aplicación) e identificador de sesión option.

-Información de invite: por cada establecimiento de sesión VS que lleve a cabo el Gateway, se creara un registro al respecto, se actualizará durante el proceso y se borrara cuando la sesión termine: username de la aplicación, identificador de sesión invite, identificador de sesión de video, SIPURI del contacto RCS-e remoto, SDP recibido y tiempo de expiración antes de la cancelación de sesión.

-Información de sesión de video: por cada sesión de video abierta en el servidor de video, tras el establecimiento previo, se creará una entrada en base de datos con la información asociada a la sesión, en base a la cual el Gateway actuará: username de la aplicación, puerto local para flujo RTP, flujo entrante o saliente, información de almacenamiento del video, url de publicación RTSP, url de publicación RTMP, identificador de sesión de video, etc.

- **Contactos/usuarios de la aplicación**

Para cada aplicación adscrita al Gateway, se mantendrá en base de datos su lista de contactos (usuarios de la aplicación). Por cada contacto se crea una entrada en base de datos, ya sea esta introducida directamente por el administrador del sistema o por el servidor de control, previa solicitud de la aplicación por invocación de la acción pertinente. La información de contacto que el sistema manejara cuenta con los siguientes campos:

-Identificador de la aplicación usuaria.

-SIPURI del contacto/usuario de la aplicación.

- Nombre identificativo del contacto.

-Capacidad de video-share activa en el contacto.

-Fecha de la última actualización de la capacidad VS

- **Parámetros de configuración general del sistema.**

El Gateway RCS-e VS cuenta con una tabla en base de datos con información común a todas las aplicaciones y con parámetros de configuración general del sistema. Algunos de los campos más importantes son:

-El identificador de agente de usuario SIP que se incluirá en todos los mensajes SIP enviados desde el Gateway. Puerto y host empleado para señalización SIP en el servidor de control y señalización, serán incluidos en la cabera *Contatc* de los mensajes SIP.

- Localización tanto del servidor de video como del repositorio de video, incluye el host y puerto para la comunicación del servidor de control y señalización con éstos.

-Rango de puertos del servidor de video empleados para las conexiones RTP/RTCP y puertos desde donde publicará los flujos RTSP y RTMP.

-Directorio local donde se almacenará de forma temporal los ficheros de video, durante la transmisión y recepción.

3.3.2 Servidor de control y señalización

El servidor de control y señalización cuenta con una serie de módulos dedicados, que se ocupan de la comunicación con cada una de las entidades participes (aplicaciones, usuarios RCS-e, repositorio de video y servidor de video) a través de las respectivas interfaces. Estos módulos, de procesamiento muy ligero, ceden la lógica de negocio a los módulos centrales Services y SIPServices, que se encargan del control y coordinación de todos ellos. Interactuando con el gestor de base de datos, Services implementará la lógica de negocio del lado de las aplicaciones, mientras que SIPServices se encarga de controlar la señalización SIP y las transmisiones de video.

- **API del sistema**

Este módulo se encarga de establecer la comunicación con las aplicaciones usuarias del Gateway:

Recibe los mensajes procedentes de las aplicaciones que solicitan alguna de las acciones del servicio e informan al módulo Services para que inicie el procesamiento requerido.

Mediante solicitudes del módulo Services se encarga de mandar notificaciones a las aplicaciones para informarles acerca de eventos producidos a nivel SIP y RTP, el progreso de las acciones y el resultado de éstas.

- **Services**

Este módulo gestiona las acciones del API invocadas por las aplicaciones usuarias. A través del gestor de base de datos realiza las comprobaciones oportunas acerca del usuario invocante y el resto de datos pasados como parámetros y luego solicita al módulo SIPServices que realice las funciones de señalización asociadas a dicha acción.

Por otro lado, cuando se produzca algún evento a nivel SIP o RTP, tal como una invitación recibida de un usuario remoto, finalización del establecimiento de sesión, finalización de transmisión RTP, etc., el módulo SIPServices se lo pondrá en conocimiento, entonces solicitará al gestor de datos la URL asociada a la aplicación donde ésta recibe las notificaciones, e invocará la función correspondiente del API.

Adicionalmente a lo anterior, otras funciones de este módulo son:

-El control del modulo de acceso a repositorio. Cuando recibe una solicitud de envío de video almacenado en fichero, antes de solicitar el establecimiento de sesión a SIPservices, obtendrá el fichero fuente de la transmisión mediante petición a este módulo. Para el caso de recepción de video, antes de notificar la finalización de recepción a la aplicación, solicitara a este módulo el almacenamiento del fichero grabado.

-Gestión de la listas de contactos. Interactúa con el gestor de datos para añadir, borrar, obtener y actualizar contactos de la lista, ya sea por solicitud de una acción invocada por

la aplicación, o porque SIPServices le comunique algún evento producido por el intercambio de SIP OPTIONS.

- **SIPServices**

Recibirá solicitudes del módulo Services para realizar un registro de aplicación, para el intercambio de OPTIONS con uno o varios de los contactos de una aplicación o para el establecimiento de una sesión de video. Este módulo se encarga de secuenciar todas las operaciones necesarias, interactuando con el Gestor de SDP´s, la pila SIP y el módulo de acceso al servidor de video, mediante peticiones y eventos. Para ello empleará la información pasada por el módulo Services procedente de la aplicación y solicitará al gestor de datos información del perfil SIP de la aplicación invocante así como de su perfil de medias.

SIPServices, apoyándose en el gestor de datos, empleará persistencia para el control de sus operaciones. Por cada solicitud ya sea procedente del módulo Services o por iniciativa de un usuario RCS-e a través del interfaz SIP, se creará una entrada en base de datos con información de registro, información de options o información de invite, asociándose ésta al usuario/aplicación correspondiente. Esta información será accedida, actualizada y borrada conforme se ejecuten las funciones y se produzcan eventos asociados al mismo, manteniendo así el estado de procesamiento y permitiendo dar servicio a múltiples usuarios/aplicaciones en paralelo. Del mismo modo, también se realizara el control de las sesiones de video abiertas por el servidor de video, manteniendo el estado de la misma mediante una entrada en base de datos.

- **Pila SIP**

Este módulo es empleado para la generación y envío de mensajes SIP bajo el control del módulo SIPServices. La información que conforman las cabeceras de los mensajes SIP será proporcionada por el módulo SIPServices a partir del perfil SIP del usuario, obtenido de base de datos.

Por otro lado, la pila SIP recibirá todos los mensajes y respuestas SIP procedentes del core, tras un parseo de las cabeceras, informará al módulo SIPServices a modo de evento para que éste se encargue de la gestiones pertinentes.

- **Gestor de medias**

Este es un módulo de utilidad que contiene todas las funciones necesarias para llevar a cabo el negociado de medias según el requerimiento 8 visto en el apartado anterior:

-Generación de mensajes SDP a partir del perfil de medias asociado a cada aplicación.

-Parseo e interpretación de los mensajes SDP recibidos, procedentes de los usuarios RCS-e remotos.

-Intersección de parámetros a partir de SDP recibido del usuario y el SDP propio de la aplicación.

Estas funciones son accedidas desde SIPServices durante el establecimiento de una sesión VS. SipServices le proporcionará el perfil de medias del usuario/aplicación previa petición al gestor de datos, así como los SDP contenidos en los mensajes SIP INVITE y respuestas 200Ok, recibidos por la pila SIP. El gestor de medias en última instancia le devolverá el mensaje SDP resultado de la negociación y la información que será posteriormente provisionada al servidor de video para el trascodificado y envío RTP de video.

Por otro lado, el Gestor de medias implementa internamente el mecanismo de gestión de puerto de las conexiones RTP/RTCP abiertas en el servidor de video.

- Se le proporciona el rango de puertos que el servidor de video empleará para envío de video.

-En orden ascendente asignará puertos a las conexiones indicándolo en la cabecera de media (m=) del SDP. Cuando llega al final del rango volverá al principio, tomando los puertos liberados, y así sucesivamente.

-Cuando una sesión de VS finalice SIPServices le hará llegar una notificación para que esos puertos pueda volver a ser asignado posteriormente.

- **Acceso a Servidor de Video**

Este módulo es empleado por SIPServices para la comunicación con el servidor de video. A través de este módulo se le pasará los parámetros de medias negociados durante el establecimiento, se le indicará o proporcionará la fuente de video o el destino de éste (fichero de video o url de tipo rtsp/rtmp) y en última instancia se solicitará que inicie la transmisión o recepción RTP. Del mismo modo cuando la aplicación decida terminar la sesión, o se reciba un mensaje SIP BYE del usuario remoto, SIPServices indicará a al módulo de acceso al SV que solicite finalización de transmisión/recepción.

Este módulo, adicionalmente al envío de peticiones, se encarga de recibir las notificaciones generadas por el servidor de video, cuando éste detecta algún evento a nivel RTP: finalización de transmisión o fallo en la transmisión.. Estas notificaciones serán pasadas al módulo SIPServices, que se encargará de realizar las gestiones pertinentes: inicio de cierre de sesión, envío de mensajes de error, propagación hacia la aplicación, etc.

- **Acceso a repositorio**

Este módulo actúa como cliente del repositorio de video, implementando dos tipos de peticiones:

-PUT, para solicitar al repositorio almacenamiento de fichero de video. La petición contiene el fichero a almacenar como un array de bytes, el identificador del espacio de

almacenamiento donde se guardará, y el nombre del fichero con el que este será identificado (<nombre_fichero>.mp4). El fichero de video grabado, procedente del servidor de video, tras pasar por el modulo de acceso al SV y SIPServices, le será proporcionado por Services. El nombre del fichero y el identificador del espacio de almacenamiento, también proporcionados por Services, será obtenido de los parámetros de la acción invocada por la aplicación. El repositorio como respuesta a esta petición devolverá la URL donde se ha almacenado el fichero.

-GET, para solicitar al repositorio uno de los videos almacenados en éste. En este caso la petición sólo contiene la URL donde se localiza el fichero dentro del repositorio, siendo éste devuelto como un array de bytes en respuesta a la petición. La URL, previamente pasada como parámetro en la invocación de la acción, será proporcionada por Services, y el fichero de video devuelto será llevado hasta el servidor de video, a través de Services, SIPServices y el modulo de acceso al SV, para que este sea empleado como fuente en la transmisión.

- **Gestor de base de datos.**

Se encarga de interactuar con la base de datos, donde se almacena toda la estructura de datos: obtención, borrado y actualización de datos de las tablas.

Lleva a cabo el mapeo objeto-relacional, los datos almacenados en tablas en base de datos son llevados a objetos Java y viceversa, de forma que puedan ser empleados por los módulos encargados de implementar la lógica del negocio (programados en lenguaje Java).

3.3.3 Servidor de video.

El servidor de video será implementado como un módulo separado, pudiendo ser ejecutado en otra máquina. Este módulo está diseñado en base a una arquitectura de pipelines o tuberías de procesamiento, donde un flujo de datos es transformado en varias fases secuenciales.

En nuestro caso, para transmisión, el flujo de datos es el video de un fichero o flujo RTSP/RTMP, y el procesamiento sobre éste será el requerido para proporcionar un flujo de paquetes RTP, a través de la conexión UDP, con el usuario destino. En recepción el procedimiento será justamente el inverso. En los siguientes diagramas se representa el procesamiento requerido para cada tipo de sesión VS:

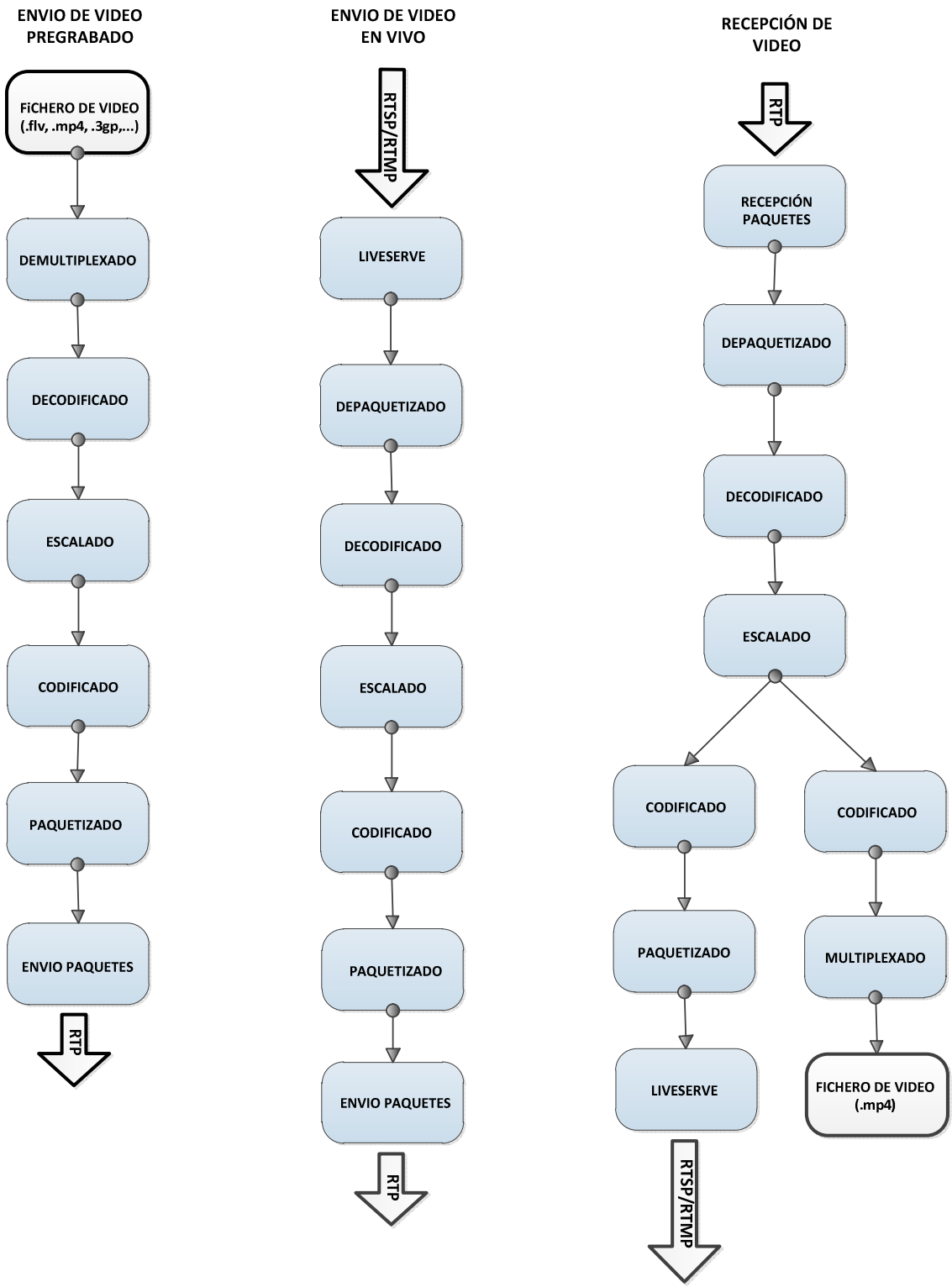


Figura 29. Procesamiento de video.

- **Envío de video pregrabado**

La fuente de la transmisión es un fichero donde está almacenada la pista de video (conjunto de frames) en alguno de los formatos soportados. Mediante demultiplexado obtenemos la pista de video y descartamos las pistas de audio o subtitulos si estas coexisten con el video. Después se decodifica los datos que conforman esta pista quedándonos con un stream de video en bruto. Sobre este stream se lleva a cabo un proceso de escalado, donde se ajusta el tamaño de lo frames del video que se emitirá así como el framerate de éste, según lo negociado durante el establecimiento. Posteriormente se vuelve a codificar a alguno de los códecs recomendados para VS que previamente también fue negociado (H263 o H264). Finalmente se paquetizan el video en RTP de acuerdo al profile que se requiera y se empiezan a enviar los paquetes por la conexión UDP.

- **Envío de video en vivo**

Para este caso se emplea las librerías proporcionadas por el proyecto Liveserve, para integrar la funcionalidad de éste en el servidor de video, como un elemento más del procesado. Liveserve actúa como cliente, conectándose a un servidor de video en streaming, empleando el protocolo RTSP o RTMP. Liveserve funciona como un conversor de protocolos proporcionando a su salida un flujo RTP.

Liveserve sólo soporta video codificado en H263 y Sorensen (variante para flash), por lo que se hace necesario el depaquetizado y decodificado del flujo, de manera que se pueda trascodificar a H264 para su envío. El resto del procesado se lleva a cabo de igual forma que para el caso anterior.

- **Recepción de video**

En recepción, el proceso es el inverso. Conforme se reciben los paquetes RTP, se realiza un depaquetizado para obtener los datos de video contenidos en éstos. Luego se procede a decodificar la información contenida en estos datos, obteniendo un flujo de video en bruto, de forma que se pueda convertir al formato y códec deseado.

Como ya vimos, en recepción hay dos posibilidades: grabado de video en fichero o retransmisión en streaming a las aplicaciones mediante RTMP y RTSP. En el primer caso, se codifica el video empleando el códec H264 y luego se utiliza un multiplexador que nos permite encapsularlo en formato mp4 dentro de un fichero. En el segundo caso, se empleará Liveserve como servidor de video bajo demanda, para ello se requiere proporcionarle a su entrada un flujo RTP, con video en H263 o Sorensen, por lo que se lleva a cabo el codificado y el paquetizado de acuerdo a ello.

Adicionalmente al procesado del video, el servidor se encarga de gestionar y enviar los paquetes RTCP a través de la conexión previamente establecida para ello, esto se lleva a cabo de forma paralela al envío y recepción de paquetes RTP.

El servidor de video, por cada sesión VS establecida por el Gateway, requerirá de un pipeline de procesamiento que mantendrá hasta que la sesión termine. De esta forma habrá tantos pipelines funcionando en paralelo como conexiones RTP/RTCP haya abiertas.

El servidor de video proporciona de cara al servidor de control y señalización una interfaz remota para la creación, gestión, activación y desactivación de pipelines. A través de ésta interfaz recibirá todos los datos que requiere: puertos a través de los que enviará o recibirá los paquetes, parámetros de escalado y framerate, códec y bitrate para codificación y decodificación, destino del video (fichero o url de tipo RTSP o RTMP), url de tipo RTSP o RTMP donde conectarse el Liveserve, etc. Adicionalmente, el servidor de video, empleará dicha interfaz para enviar notificaciones al servidor de control y señalización, ante eventos que se produzcan durante el procesado y la tansmisión del video.

Capítulo 4

Implementación del sistema

En el capítulo anterior se presentó el sistema Gateway RCS-e VS y se explicó como funciona a partir de la funcionalidad que aporta cada uno de los módulos que componen su arquitectura. Aquí trataremos de explicar como han sido implementados los distintos módulos, que técnicas y tecnologías han sido empleadas para desarrollar la funcionalidad de cada módulo y cuál ha sido el modelado software aplicado a los mismos.

La implementación de un sistema de estas características requiere de la utilización de numerosas herramientas y tecnologías. En el primer apartado de este capítulo se exponen las tecnologías bases elegidas para el desarrollo software del sistema. En el segundo y tercer apartado veremos cómo han sido aplicadas estas tecnologías para la implementación del servidor de control y señalización y el servidor de video, respectivamente. En estos dos apartados, partiendo de la arquitectura modular expuesta en el capítulo anterior, recorreremos cada uno de los módulos viendo el modelado software que se ha realizado en cada caso y aportando algunos detalles acerca de la codificación.

4.1 Herramientas y tecnologías

De acuerdo a los requerimientos del Gateway, se han elegido tres tecnologías como pilares sobre los que se realizará la implementación del sistema:

-La plataforma de programación Java para aplicaciones empresariales JEE, facilita sobremanera el desarrollo de sistemas tan complejos como el nuestro. Su modelo de componentes y sus numerosas APIs, nos permitirán gestionar de manera eficiente la información asociada a sesiones, aplicaciones, y usuarios de las mismas, y nos permitirá llevar el servicio hasta las aplicaciones.

-Los componentes Sip Servlets de Java y la plataforma Mobicents Sip Servlets. A los componentes ya provistos por EJB, añadiremos Sip Servlets, diseñados específicamente para el desarrollo de aplicaciones que requieran comunicación SIP, como es nuestro caso. Éstos requieren de contenedores de SipServlets para ser ejecutados, en nuestro caso emplearemos el servidor Mobicents SipServlets que al estar implementado sobre la base de un servidor de aplicaciones JEE, JBoss, facilitará la integración de estos Sip Servlets con el resto de componentes JEE que constituyen nuestro sistema.

-El framework de Gstreamer será la base para la implementación del servidor de video. Gstreamer ofrece una serie de elementos que nos permitirá, entre otros, enviar y recibir video RTP para cada una de las sesiones VS establecidas. El acceso a estos elementos se realiza a través del API de Gstreamer para aplicaciones audiovisuales.

4.1.1 Java Enterprise Edition

Java Edition Enterprise [45] define el estándar para el desarrollo y ejecución de aplicaciones empresariales distribuidas, con arquitectura en niveles.

JEE simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados que se ejecutan dentro un servidor de aplicaciones. Un servidor de aplicaciones sirve como contenedor de componentes, proveyendo un completo conjunto de servicios a estos componentes y manejando así muchas de las funciones de la aplicación a bajo nivel de forma automática, sin necesidad de una programación compleja (concurrencia, gestión de estado, acceso a recursos, etc.). En la especificación hay definidos tres tipos de componentes:

- Componentes de cliente: aplicaciones instaladas en el cliente y los applets.
- Componentes Web o de Red: Java Servlets y Java Pages (JSPs).
- Componentes de Negocio: Enterprise Java Beans.

Java EE es una especificación que incluye varias APIs concretas y define la coordinación entre ellas para que el desarrollador pueda crear sus aplicaciones de una manera sencilla y centrada en la lógica propia de la aplicación. Una visión global de la arquitectura de JEE puede verse en la siguiente figura:

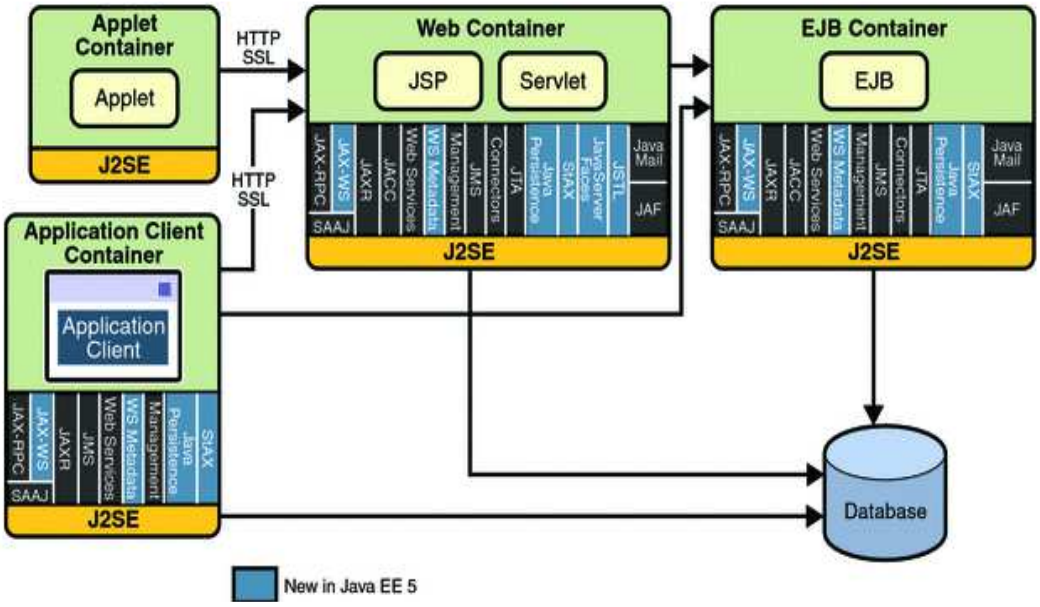


Figura 30. Arquitectura de JEE

JEE se estructura, aunque no de forma rigurosa, en cuatro niveles:

-Nivel cliente: navegadores, aplicaciones standalone, otros servidores (en diferentes máquinas que el servidor JEE).

-Nivel web: Gestiona la interacción entre clientes web y la capa de negocio (JSP, servlets,etc).

-Nivel de negocio: provee la lógica de negocio de la aplicación o el servicio (EJB).

-Nivel de datos: la base de datos donde se aloja la estructura de datos empleada por la aplicación o el servicio.

En nuestro sistema, el nivel cliente lo constituirán las aplicaciones usuarias adscritas al Gateway, que estarán ejecutandose en otro servidor, mientras que la capa de red o web, la capa de negocio y la capa de datos, ya formando parte del Gateway, quedan integradas en el servidor de control y señalización. Las dos primeras son implementadas sobre un servidor de aplicaciones Jboss [46], que funciona como contenedor de EJBs, llevando además integrado el contenedor de servlets Tomcat [47]. Para la capa de datos emplearemos una base de datos relacional postgresSQL [48] .

El API del sistema encargado de la interacción con las aplicaciones clientes para llevar hasta éstas el servicio VS, constituirá la capa de red de nuestra arquitectura. Los módulos tales como Services, SIPServices y el sistema gestor de datos, sobre los que se implementa la lógica del servicio ofrecido, formarán parte de la capa de negocio.

En los siguientes puntos se describe de forma breve las tecnologías proporcionadas por JEE que se han empleado para la implementación del sistema.

- **Enterprise Java Beans [45] [49].**

Escritos en el lenguaje de programación Java, un Enterprise Bean es un componente software del lado del servidor que encapsula la lógica del negocio de una aplicación o servicio. La lógica del negocio es el código que satisface el propósito de la aplicación o el servicio.

Los EJB simplifican el desarrollo de grandes aplicaciones distribuidas. El contenedor de EJBs proporciona servicios a nivel de sistema a los Enterprise Bean, por lo que el desarrollador de estos componentes puede concentrarse solamente en resolver los problemas de la lógica del negocio. Por su parte, el contenedor de EJBs es responsable de los servicios a nivel de sistema tales como el manejo de transacciones y seguridad. Se recomienda el uso de esta tecnología cuando:

-La aplicación deba ser escalable, por ejemplo, si existe un incremento en el número de usuarios de la aplicación tal vez sea necesario distribuir los componentes que forman la aplicación entre varios ordenadores. Esta tecnología permite que la ubicación de los componentes sea transparente para sus clientes.

- Las transacciones deban garantizar la integridad en los datos. Los EJB soportan el manejo de transacciones.

- La aplicación deba tener una diversidad de clientes. Con unas pocas líneas de código, los clientes pueden localizar y usar este tipo de componentes.

Existen dos tipos de Enterprise Java Beans:

-**Sesión**: gestionan el flujo de la información en el servidor. Realizan una tarea para un cliente; opcionalmente pueden implementar un servicio web. Dentro de los EJB de sesión hay dos tipos:

Con estado (statefull): En un EJB de sesión con estado, las variables de instancia del bean almacenan datos específicos obtenidos durante la conexión con el cliente. Cada bean de sesión con estado, por tanto, almacena el estado conversacional de un cliente que interactúa con el bean. Este estado conversacional se modifica conforme el cliente va realizando llamadas a los métodos de negocio del bean.

Sin estado (stateless): Los beans de sesión sin estado son objetos distribuidos que carecen de estado asociado, permitiendo por tanto que se los acceda concurrentemente. No se garantiza que los contenidos de las variables de instancia se conserven entre llamadas al método

-**Dirigido por mensajes**: Actúan como un escuchador para un tipo de mensaje en particular, tal como la API del servicio de mensajes de Java.

Un EJB está formado por unas interfaces donde se definen los métodos o funciones que realiza, y una clase donde son implementados estos métodos. Las interfaces pueden ser locales o remotas y dan acceso a los clientes del EJB, que se limitan a invocar los métodos definidos en estas interfaces, quedando oculto para ellos aspectos de la implementación. Un cliente puede ser otro EJB corriendo en el mismo contenedor, en este caso la interacción se realiza a través de la interfaz local. Si el cliente esta implementado fuera del contenedor, la invocación de métodos se realiza sobre la interfaz remota. Existen diferentes tipos de clientes que emplean diferentes tecnologías para acceder al servicio prestado por un EJB desde el exterior: clientes web services empleando el protocolo de comunicación SOAP (Simple Object Access Protocol) [49], aplicaciones Java que emplean invocación remota de métodos (RMI) [52] o clientes CORBA [53].

- **JDBC y Entidades JPA**

Java Database Connnectivity, (JDBC) [54], es un API que permite la conectividad entre el lenguaje de programación Java y cualquier base de datos relacional, empleando el dialecto SQL del modelo de base de datos que se utilice. Se trata de un API a nivel de llamada para base de datos que emplean SQL.

El API JDBC se compone de una colección de interfaces Java y métodos de gestión para manejar conexiones hacia un modelo de base de datos en particular. El usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de ahí puede realizar cualquier tipo de tareas con la base de datos a las que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en base de datos, etc.

Java Persistence API (JPA) [45], es la especificación de persistencia desarrollada para la plataforma Java. Su objetivo fundamental es estandarizar la forma en que funcionan los sistemas que permiten hacer mapeos objeto-relacional. Estos mapeos consisten básicamente en transformar la información modelada en un diseño orientado a objetos para ser persistido por un sistema de base de datos relacional. Una clase entidad JPA representa una tabla en base de datos relacional y cada una de las instancias de esta clase representa una fila de una tabla

De esta forma, se mantiene las ventajas de la orientación a objetos mientras que se trabaja con sistemas de bases de datos relacionales potentes y contrastados.

- **Web Services y JAX-WS**

Un Web Services [45] [55] es un mecanismo o sistema software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web

para intercambiar datos en redes de ordenadores como Internet. Los estándares empleados son los siguientes:

-**XML** (Extensible Markup Language) [51]: Es el formato estándar para los datos que se vayan a intercambiar.

-SOAP (Simple Object Access Protocol) [50]: Protocolo sobre el que se establece el intercambio. Generalmente los mensajes SOAP, son enviados empleado peticiones HTTP, que encapsulan en su cuerpo los datos en formato XML a intercambiar

-WSDL (Web Services Description Language): Es el lenguaje de la interfaz pública para los servicios web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios web.

-UDDI (Universal Description, Discovery and Integration) : Protocolo para publicar la información de los servicios web. Permite comprobar qué servicios web están disponibles

JAX-WS [45] es el API de Java para XML Web Services. Está tecnología permite construir web services y clientes que se puedan comunicar usando XML. JAX-WS permite a los desarrolladores escribir mensajes para invocación remota de las operaciones que conforman los web services, empleando lenguaje de programación Java.

En JAX-WS la invocación de operaciones de un servicio web es definida por el protocolo SOAP empleando el descriptor WSDL. El WSDL define la estructura, las reglas de codificación y los datos requeridos para poder invocar dichas operaciones. Estas llamadas serán transmitidas como mensajes SOAP (ficheros XML) sobre HTTP.
Aunque los mensajes SOAP son complejos el API JAX-WS esconde esta complejidad al desarrollador de aplicaciones. En el servidor, el desarrollador especifica las operaciones mediante la definición de métodos en una interfaz escrita en lenguaje de programación Java. El desarrollador además codifica una o más clases que implementan esos métodos. Del lado del cliente la codificación también es muy fácil. Un cliente crea un proxy (un objeto local que representa el servicio) y luego simplemente invoca los métodos sobre el proxy. Con JAX-WS el desarrollador no genera ni parsea mensajes SOAP. El “runtime system” de JAX-WS es quien convierte las llamadas y respuestas del API implementado en lenguaje Java a mensajes SOAP y viceversa

- **RMI**

Java Remote Method Invocation (RMI) [52] es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI.

A través de RMI un programa Java puede exportar un objeto, con lo que dicho objeto estará accesible a través de la red, quedando el programa a la espera de peticiones en un

puerto TCP. A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

La invocación se compone de los siguientes pasos:

- Encapsulado (marshalling) de los parámetros
- Invocación del método (del cliente sobre el servidor). El invocador se queda esperando una respuesta.
- Al terminar la ejecución, el servidor serializa el valor de retorno (si lo hay) y lo envía al cliente.
- El código cliente recibe la respuesta y continúa como si la invocación hubiera sido local.

4.1.2 Mobicents Sip Servlets

- **SIP Servlet**

Un Sip Servlet [56] es un componente de aplicaciones Java administrado por un contenedor de SIP Servlets para realizar funciones de señalización SIP. Un SIP Servlet interactúa con un cliente SIP intercambiando peticiones y respuestas a través de un contenedor de servlets.

La especificación SIP Servlet v1.1 (JSR289), proporciona una interfaz de programación de aplicaciones (API) similar a los servlets HTTP, que proporcione un modelo de programación SIP fácil de utilizar. Algunos de los objetivos del API son:

-Señalización SIP: proporcionar a las aplicaciones un completo conjunto de acciones de señalización SIP, incluyendo el soporte necesario para que puedan actuar como UAS, UAC, o proxies.

-Aplicaciones convergentes: posibilidad de que los contenedores puedan soportar aplicaciones convergentes, es decir, aplicaciones que abarcan múltiples protocolos e interfaces, por ejemplo, Internet, telefonía y otras interfaces de Java EE

-Desarrollo de aplicaciones por terceros: un descriptor de despliegue XML se utiliza para que el desarrollador pueda incluir información de la aplicación para el despliegue.

-Composición de aplicaciones: se puede ejecutar varias aplicaciones sobre la misma petición/respuesta ya sean entrantes o salientes. Cada aplicación tiene sus propias reglas y se ejecuta independiente a las otras, de forma bien definida y ordenada.

El API de servlets SIP difiere de muchas maneras de los servlets HTTP debido a que el protocolo es diferente. Mientras que SIP es un protocolo de petición y respuesta, no necesariamente existe una sola respuesta para cada petición individual. Esta complejidad y la necesidad de obtener una solución de alto rendimiento significan que resulta más fácil hacer que los servlets SIP sean asíncronos de forma nativa.

Cada método de petición, que debe soportar la especificación, tiene un método doxxx del mismo modo que HTTP. En SIP, existen los métodos doInvite, doAck, doOptions, doBye, doCancel, doRegister, doSubscribe, doNotify, doMessage, doInfo y doPrack para cada petición SIP.

Los servlets SIP incluyen las respuestas doProvisionalResponse, doSuccessResponse, doRedirectResponse y doErrorResponse. Específicamente, se utilizan las respuestas provisionales (respuestas 1xx) para indicar el estado, las respuestas satisfactorias (respuestas 2xx) para indicar que una transacción se realizado satisfactoriamente, las respuestas de redirección (respuestas 3xx) para redirigir al cliente a un recurso o entidad que se ha movido y las respuestas de error (respuestas 4xx, 5xx y 6xx) para indicar una anomalía o una condición de error específica. Estos tipos de mensajes de respuesta son similares a HTTP, pero debido a que el modelo de programación del servlet SIP incluye un modelo de programación de cliente es necesario que las respuestas también se manejen de modo programado.

- **Servidor Mobicents Sip Servlets [57] [58]**

El contenedor de servlets es una parte de un servidor de aplicaciones que proporciona los servicios de red sobre los cuales las peticiones y respuestas son recibidas y enviadas. Decide qué aplicaciones invocar y en qué orden. Un contenedor de servlets también contiene y gestiona los servlets través de su ciclo de vida.

Un contenedor de SIP servlets gestiona la red de puntos de escucha para el tráfico SIP entrante (un punto de escucha es una combinación de protocolo de transporte, la dirección IP y número de puerto). La especificación SIP requiere que todos los elementos de SIP puedan soportar UDP, TCP, y, opcionalmente TLS y otros medios de transporte.

La especificación distinguen tres tipos de contenedores respecto a la posibilidad del desarrollo de aplicaciones convergentes: contenedores standalone SIP Servlets, contenedores para la convergencia de HTTP y SIP servlets y contenedores para la compatibilidad de SIP y Java EE. Este último caso será el de interés para nosotros, ya que nos permitirá emplear conjuntamente tecnologías tales como EJB o Web Services con las que podemos llevar el servicio hasta las aplicaciones usuarias del Gateway y el API de SIP Servlet con el que estableceremos la comunicación SIP con los clientes RCS-e.

Mobicents Sip Servlets es una plataforma sobre la cual se puede desarrollar y desplegar de forma portable y distribuible servicios SIP y JEE. El servidor Mobicents Sip Servlets es una implementación certificada de la especificación SIP Servlet v1.1 (JSR 289 Spec) que puede correr ya sea sobre el servidor de aplicaciones JBoss [46] o sobre el contenedor de servlets HTTP Tomcat [47]. Mobicents para JBoss tiene como objetivo proporcionar una completa interoperabilidad entre SIP Servlets y JEE de forma que las aplicaciones puedan explotar las fortalezas de ambas.

4.1.3 Gstreamer

GStreamer [59] es un framework multimedia libre y multiplataforma, escrito en lenguaje de programación C, que permite crear aplicaciones audiovisuales. La biblioteca de Gstreamer permite la reproducción de sonido y vídeo, grabar una entrada de múltiples fuentes y editar contenido multimedia.

GStreamer proporciona una arquitectura flexible en la que se procesa el contenido multimedia a través de una pipeline de elementos:

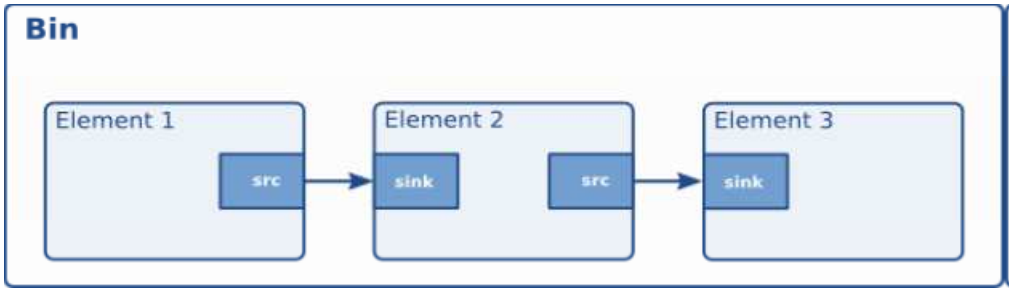


Figura 31. Arquitectura de pipeline de elementos

-Elemento: es la parte fundamental dentro de la clase de objetos en Gstreamer. Nos permite crear una cadena de elementos enlazados entre si y lograr que los datos fluyan por ella. Un elemento tiene funciones específicas, como leer datos de un archivo, decodificar los datos o enviarlos a una tarjeta de sonido (u otro dispositivo). Colocando en una cadena distintos elementos, podríamos realizar tareas específicas, como reproducción o captura multimedia. Gstreamer provee de una amplia colección de elementos.

- **Bin:** es un contenedor para un conjunto de elementos. Su utilidad está en, por ejemplo, cambiar el estado de todos los elementos de un bin cambiando solo el estado de éste bin contenedor

-Pipelines: son bin de mayor nivel que pueden agrupar además de elementos otros bins.

-Pads: son usados para negociar enlaces y flujo de datos entre elementos de Gstreamer. Pueden ser de salida (source) o entrada (sink).

La arquitectura de Gstreamer permite incorporar complementos o plugins que añadan codificadores, decodificadores y todo tipo de filtro de contenidos. Los desarrolladores de terceras partes pueden proporcionar complementos para Gstreamer que estarán disponibles automáticamente para todas las aplicaciones que usen Gstreamer. Los complementos pueden proporcionar soporte para otros formatos multimedia o proporcionar funcionalidades y efectos adicionales. Los complementos se organizan en paquetes de este modo:

-**gst-plugins-base** contiene el conjunto básico de complementos bien soportados.

-gst-plugins-good contiene el conjunto de complementos bien soportados que usan licencias preferidas (libres) por los desarrolladores de Gstreamer.

-gst-plugins-ugly contiene el conjunto de complementos bien soportados, pero que podrían tener problemas para su libre distribución.

-gst-plugins-bad contiene el conjunto de aquellos complementos menos desarrollados que no han pasado las rigurosas pruebas de calidad de los desarrolladores.

La función del core de Gstreamer es proveer un framework para plugins, flujo de datos y manejo/negociación de distintos tipos de medios. También provee una API para escribir aplicaciones. Una visión general acerca de Gstreamer se puede ver en el siguiente diagrama:

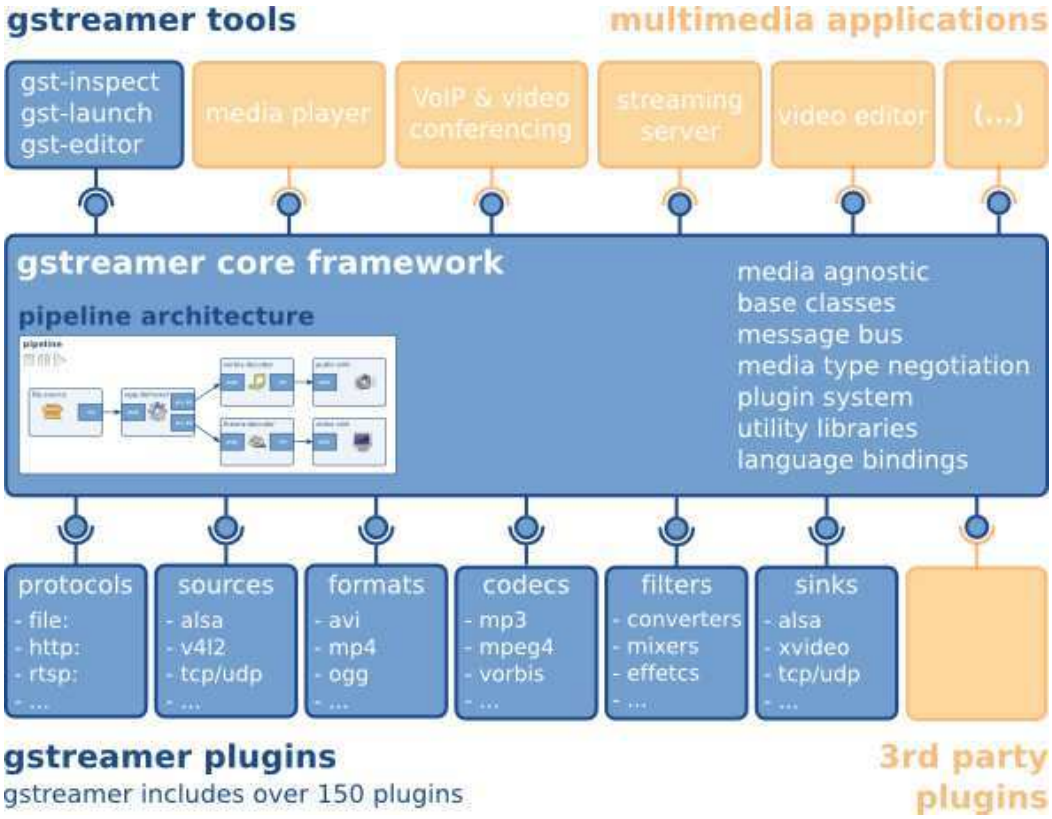


Figura 32. Visión general de Gstreamer

- **Gstreamer-Java. [60]**

El API proporcionado por Gstreamer para el desarrollo de aplicaciones y complementos, aunque escrita originariamente en C, puede ser accedido por muchos otros lenguajes de programación, tal como Python, Perl, Guile o Ruby.

Para el caso del lenguaje Java, el cual se ha elegido para el desarrollo de todo el software de nuestro sistema, existe un proyecto llamado Gstreamer-Java que provee un API en Java para acceder al framework de Gstreamer. Gstreamer-Java emplea las librerías de Java Native Access (JNA) [61] para mapear las funciones del API de Gstreamer escritas en lenguaje C a funciones propiamente Java.

Aunque el proyecto Gstreamer-Java aún está en desarrollo, se ha encontrado que el API publicado cubre los requerimientos funcionales de nuestro servidor de video.

4.1.4 Tecnologías en RCS-e VS Gateway

En el siguiente diagrama se presenta, a modo de resumen el conjunto de todas las tecnologías empleadas en el Gateway:

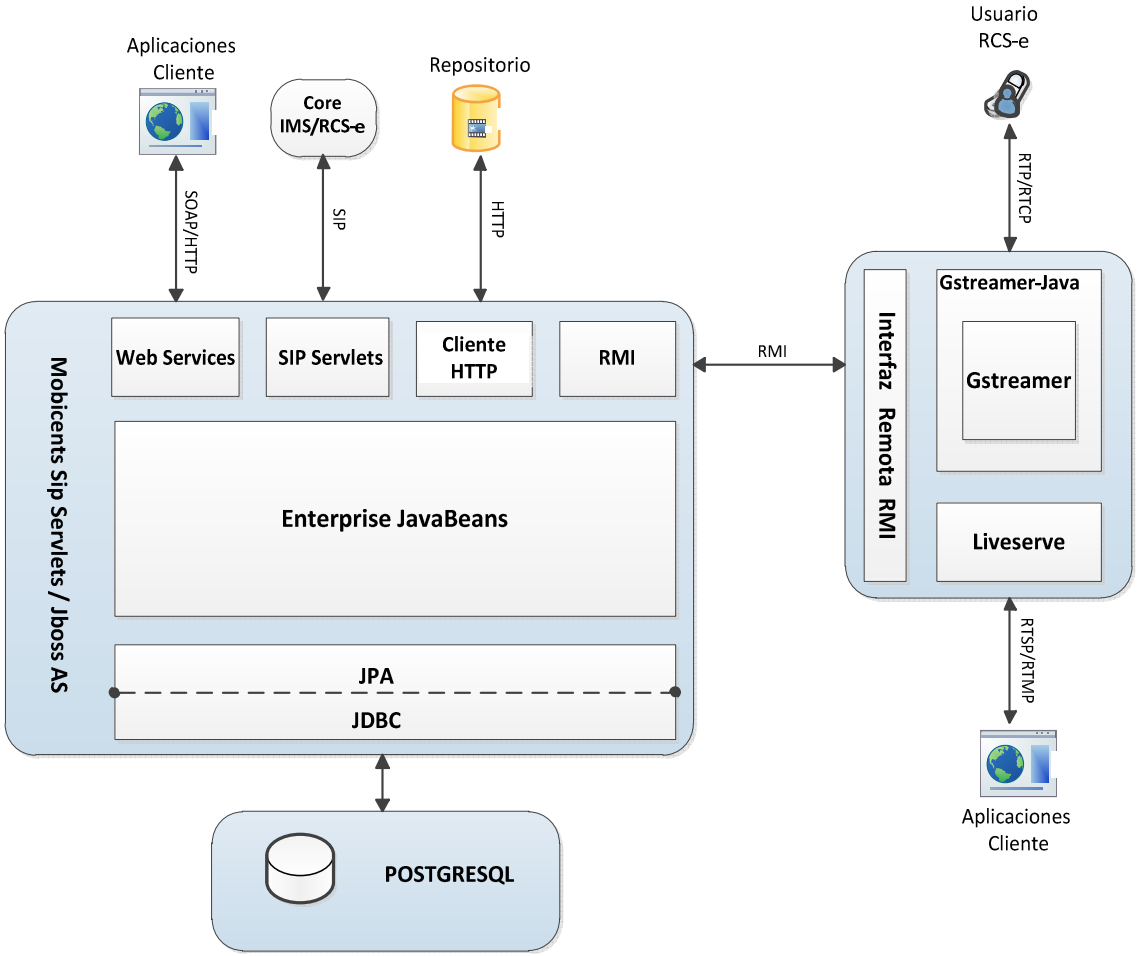


Figura 33. Tecnologías en RCS-e VS Gateway

El servidor de control y señalización se ejecutara en un servidor Mobicents Sip Servlets, incluido en el servidor de aplicaciones JBoss. A partir de las distintas tecnologías empleadas en su desarrollo, podemos distinguir tres capas:

-Una capa de datos, donde la información manejada por el sistema será alojada en una base de datos relacional postgresSQL. Para la conexión y el acceso a dicha base de datos desde el servidor, emplearemos las librerías de JBDC. Mediante entidades JPA, se realizará el mapeo objeto-relacional para poder procesar la información empleando lenguaje Java.

-Una capa de negocio, que contará con numerosos componentes EJB de sesión sin estado a través de los que se modelará los distintos módulos que componen el servidor de control y señalización. Estos EJB gestionan y procesan la información de sesiones, aplicaciones y usuarios del sistema.

-Una tercera capa a la que denominaremos capa de comunicación. Como vimos en la presentación de la arquitectura general del sistema, el servidor de control y señalización cuenta con distintos módulos para la comunicación con las diferentes entidades participes, externas al servidor. Las tecnologías, vistas anteriormente, que aplicaremos en cada caso son:

> Web Services, para la comunicación con las aplicaciones. Mediante intercambios de mensajes SOAP sobre HTTP las aplicaciones acceden a las acciones del API y el servidor les hace llegar las notificaciones a éstas.
>
> SipServlets, para la comunicación con los usuarios RCS-e a través del core IMS/RCS. Se envíaran y recibirán mensajes SIP, actúando como cliente y servidor SIP, para registro, establecimiento VS e intercambio de OPTIONS.
>
> Un cliente HTTP, para solicitar almacenamiento y recuperación de fichero al repositorio de video mediante peticiones HTTP.
>
> RMI será el mecanismo empleado para la interacción con el servidor de video, a tenor de que ambos son codificados en lenguaje Java. Mediante invocación remota el servidor de control y señalización controlará las transmisiones de video.

El servidor de video es implementado y ejecutado como módulo externo al servidor de control y señalización. La implementación se ha realizado en base al framework de Gstreamer empleando su arquitectura de pipelines de procesamiento y haciendo uso de los distintos complementos que proporciona, a los que accederemos empleando el API de Gstreamer-Java. El servidor de video será completado con la integración del servidor Liveserve, para sesiones RTSP/RTMP, y con una interfaz RMI mediante la cual se comunicará con el servidor de control y señalización.

4.2 Servidor de control y señalización

Los módulos que componen el servidor de control y señalización han sido implementados en paquetes software, donde se incluyen los componentes (EJB, SipServlets, WebServices, Entidades JPA) sobre los que se han modelado su funcionalidad. Todos estos paquetes, junto con los ficheros de configuración y despliegue, son incluidos dentro de un fichero EAR [46], para su despliegue en el servidor Mobicents Sip Servlets, el cual realiza funciones de contenedor de EJB, contenedor Web, y contenedor de SipServlets.

En los siguientes subapartados, para cada módulo, se verán los componentes que los conforman y se detallarán aspectos reseñables acerca de la codificación, configuración y empleo de APIs.

4.2.1 Sistema gestor de datos

- **Configuración Base de datos [48].**

1. El primer paso será crear la base de datos en un servidor PostgreSQL, previamente instalado en la máquina que aloja al servidor de control y señalización, aunque podría estar en otra máquina si se desea realizar reparto de cargas.

```
sudo -u postgres createdb -O solaiemes RCSVS
```

2. Luego se crea el conjunto de todas las tablas que alojan toda la información manejada por el sistema. Para ello se ha elaborado un script SQL (DBCreation.sql), empleando el comando CREATE del lenguaje de definición de datos (DDL) proporcionado por SQL, donde se definen todas la tablas y sus campos.

```
psql -h localhost --username=solaiemes RCSVS < DBCreation.sql
```

3. El administrador del sistema introducirá la información de carácter estático, no generada durante el funcionamiento del sistema: perfiles SIP de aplicaciones, perfiles de medias de aplicaciones, así como parámetros generales de configuración. Para ello se ha creado sendos scripts SQL a modo de plantilla, donde se hace uso del comando INSERT del lenguaje de manipulación de datos (DML)

```
psql -h localhost --username=solaiemes RCSVS < INSERT_SIP_USER.sql
psql -h localhost --username=solaiemes RCSVS < INSERT_MEDIA_USER.sql
psql -h localhost --username=solaiemes RCSVS < INSERT_CONFIGVALUES.sql
```

4. Por último, se incluye en el servidor de aplicaciones MSS/Jboss, el conector JDBC de PostgreSQL (postgresql-9.1-901.jdbc4.jar) y el fichero “datasource”, donde se le indica la localización de la base de datos y parámetros requeridos para acceder a ella, tal como el username y password (rcsVS-postgre-ds.xml).

- **Modelado y gestión de los datos**

Los datos recogidos en las distintas tablas se corresponden con lo visto en el apartado 3.3.1 donde se describió el modelo de datos general del sistema. La implementación del sistema gestor de datos en el servidor de control y señalización se ha realizado en dos paquetes:

-com.solaiemes.rcsvs.model

Contiene el conjunto de entidades JPA que se han empleado para modelar la base de datos como objetos Java. Estas entidades son las responsables del mapeo Objeto/Relacional. Cada una de las clases del paquete es una entidad JPA, que se corresponden con cada una de las tablas de base de datos, cada objeto de dichas clases representan una entrada en base de datos y sus atributos se corresponden con los distintos campos de la tabla. Estas clases sólo cuentan con métodos "get" y "set" para la obtención y asignación de valores de los atributos.

-com.solaiemes.rcsvs.dao

Este paquete provee los EJBs para implementar las operaciones básicas sobre las entidades JPA que componen el sistema. La funcionalidad ofrecida por este conjunto de EJBs se corresponde de un modo genérico con el patrón DAO (Data Access Object), que oculta las tecnologías y el modo de acceso a los datos, delegando, en este caso, las operaciones concretas en el EntityManager de JPA.

Por cada tabla/entidad se ha implementado un EJB DAO encargado de su gestión (operaciones CRUD [create, read, update, delete]: acceso, altas, bajas y modificiaciones) . Todos los EJBs son sin estado y cuentan con una clase donde son implementadas las funciones para la gestión (XxxxDaoBean.java) y una interfaz local (XxxxDaoBeanLocal.java), mediante la cual, los EJB de los módulo Services y SIPServices pueden acceder a dichas funciones.

Base de datos RCSVS Tabla	**com.solaiemes.rcsvs.model** Entidad JPA	**com.solaiemes.rcsvs.dao** EJB
SIP_USERS	SipUserJPA	SipUserDaoBean
MEDIA_USERS	MediaUserJPA	MediaUserDaoBean
CONFIG_VALUES	ConfigValueJPA	ConfigValueDaoBean
REGISTER_INFO	RegisterInfoJPA	RegisterInfoDaoBean
INVITE_INFO	InviteInfoJPA	InviteInfoDaoBean
OPTION_INFO	OptionInfoJPA	OptionInfoDaoBean
CONTACTS	ContactsJPA	ContactsDaoBean
VS_SESSION_INFO	VsSessionInfoJPA	VsSessionInfoDaoBean

Tabla 3. Modelado del sistema gestor de datos

4.2.2 API del sistema.

Para la implementación del API del sistema, a través del cual el servidor de control y señalización interactúa con las aplicaciones usuarias del Gateway, se ha empleado Web Services.

En el caso de las acciones del API, éstas constituirán las operaciones de web services implementados en el servidor de control y señalización. Las aplicaciones invocarán estás operaciones/acciones de forma remota mediante envío de mensajes SOAP sobre peticiones HTTP POST, donde son encapsulados los parámetros de la acción como datos XML.

Para las notificaciones, mediante las que se establece la comunicación desde el servidor de control a la aplicación cliente, se empleará SOAP callbacks. La metodología es similar a la empleada para las acciones del API, pero en este caso el servidor hará el papel de cliente siendo quien envíe los mensajes SOAP/HTTP hacia las aplicaciones, donde se alojan los web services cuyas operaciones representan las distintas notificaciones del Gateway.

El módulo API del sistema está formado por dos paquetes. Uno donde se incluyen los web services encargados de recibir las invocaciones de las acciones y otro donde son implementados los clientes encargados de enviar la notificaciones, a modo de mensajes SOAP/HTTP, hacía las aplicaciones.

- **Acciones (com.solaiemes.rcsvs.webservices)**

Las acciones del API se han dividido en tres grupos de acuerdo a la funcionalidad a la que dan acceso y a sus requerimientos de señalización SIP: acciones para sesiones VS (establecimiento, aceptación, cierre), acciones de sesión general (registro y deregistro) y acciones para la gestión de contactos/usuarios. Cada uno de estos grupos forman las operaciones de un web service distinto.

Web Services	**VideosShareWS**	**SessionWS**	**ContacsWS**
Operaciones	-sendFileVideoshare() -closeVideoStream() -rejectVideoshare() -acceptVideoshare() -sendStreamVideoshare()	-register() -unregister()	-getContactList() -addContact() -removeContact()

Tabla 4. WebServices del API del Gateway

Empleando las librerías y etiquetas que proporciona el API JAX-WS, se ha modelado cada uno de los web services a partir de interfaces java, donde son definidas las operaciones que componen el servicio web como métodos java, y de una clase java

(XxxxWS.java) donde es codificada la implementación de dichos métodos. En nuestro caso, tras cada método/operación simplemente se hará una llamada a funciones de los EJBs que componen el módulo Services, donde se lleva a cabo la lógica de negocio que implica la acción invocada.

```
package com.solaiemes.rcsvs.webservices;

@WebService
public
import javax.ejb.EJB;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebResult;
import javax.jws.WebService;

@WebService
public class VideosShareWS {

	@EJB(mappedName = "rcsvs-ear/VsServiceBean/local")
	private VsService vsService;

	@WebMethod
	public @WebResult(name = "sendFileVideoshare")
	VsSendFileResult sendFileVideoshare(
			@WebParam(name = "username") String username,
			@WebParam(name = "destUri") String destUri,
			@WebParam(name = "subject") String subject,
			@WebParam(name = "filename") String filename,
			@WebParam(name = "contentType") String contentType,
			@WebParam(name = "url") String url

	) {
		VsSendFileResult result = new VsSendFileResult();
		vsService.sendFileVideoshare(username, destUri, subject,
filename, contentType, url);
		 result.setErrorCode(0);
		 result.setErrorDescription("");
		return result;
	}
```

Extracto 1 .Codificación de una operación de un WS mediante JAX-WS.

En el servidor de aplicaciones, los web services son desplegados en el contenedor web como si de servlets se tratara. El servidor de aplicaciones, durante el despliegue, generara los descriptores WSDL de cada uno de estos servicios a partir de las interfaces y clases implementadas empleando para ello el compilador Java-WSDL proporcionado por JAX-WS. La URL donde quedarán localizados los WSDL es configurada en el fichero de despliegue web.xml:

```
<servlet>
<servlet-name>VideoShareWS</servlet-name>
<servlet-class>com.solaiemes.rcsvs.webservices.VideoShareWS</servlet-class>
</servlet>
	<servlet-mapping>
```

```
            <servlet-name>VideoShareWS</servlet-name>
            <url-pattern>/videoshare</url-pattern>
        </servlet-mapping>
```

Extracto 2.Fragmento de fichero web.xml para el despliegue de un Web Service.

El administrador del Gateway proporcionará a los desarrolladores que hagan uso del API las URLs para poder acceder a los descriptores WSDL. A partir de la interfaces WSDL, éstos contarán con toda la información necesaria (direcciones y puertos de acceso, tipos de datos, operaciones, bindings) para poder generar y mandar sus mensajes SOAP sobre HTTP, invocando asi las operaciones (acciones) de cualquiera de los Web Service que conforman el API.

- **Notificaciones (com.solaiemes.rcsvs.notifiers)**

En el caso de las notificaciones se ha llevado a cabo una división similar a la realizada para las acciones del API, de modo que tendremos notificaciones de sesión VS, notificaciones de sesión general (resultado del registro) y notificaciones acerca de cambios en la lista de contactos/usuarios.

Las aplicaciones adscritas al Gateway que deseen recibir notificaciones del servicio deberán incluir como parte de ellas un web services por cada uno de estos grupos, cuyas operaciones se corresponderán con las respectivas notificaciones. Del lado del Gateway se implementarán tres clientes WS que se encargarán de mandar las notificaciones a modo de mensajes SOAP/HTTP invocando las distintas operaciones de los Web Services alojados en la aplicación

Cada uno de los clientes WS del Gateway se ha modelado sobre un EJB, empleando el API JAX-WS. Cada uno de los métodos del EJB se encarga de mandar una notificación en particular (invocación de una operación de un WS de la aplicación).

EJB Cliente WS	**VideoShareNotifierBean**	**SessionNotifierBean**	**ContactsNotifierBean**
Métodos (notificac.)	videoshareStarted() videoshareReceived() videshareFileStored() videoshareStopped() videoshareError()	registerSuccess() registerError() unregisterSuccess() unregisterError()	contactAdded() contactRemoved() contactChanged()

Tabla 5. Clientes WS del API del Gateway

Para la implementación de los clientes WS tendremos en cuenta las siguientes consideraciones:

-Para que el Gateway pueda dar servicio a las diferentes aplicaciones usuarias sin problemas en la comunicación SOAP, será necesario que todas cuenten con descriptores WSDL donde la estructura de los mensajes, (nombre de operaciones, orden de los parámetros, etc.) debería ser estrictamente la marcada por el Gateway. Por esto, que sea tarea nuestra generar un WSDL común para cada uno de los Web Service, siendo éstos proporcionados al desarrollador de aplicaciones (terceras partes), quien debería tener los conocimientos necesarios para la interpretación de estos descriptores, siendo el encargado de codificar la lógica del Web Service tras el interfaz WSDL según los requerimientos de su aplicación.

-Cada aplicación por tanto contará con tres WSDL cuya publicación será parte de la tarea del desarrollador, de acuerdo a la localización de su aplicación. El desarrollador deberá proporcionar al administrador del Gateway las URLs de sus descriptores, que formarán parte del aprovisionamiento de la aplicación en la base de datos del Gateway. De esta forma los clientes WS del API del Gateway cuando tengan una notificación para la aplicación en cuestión, podrán acceder a sus respectivos WSDLs para su envío.

En cuanto a los clientes WS, empleando la herramienta JAX-WS, la invocación de las operaciones de los web services alojados en las aplicaciones se realizará a través de clientes proxies (objetos locales que representa el punto de acceso al servicio) como si de métodos corrientes de Java se tratasen, a los que se le pasa como parámetros los datos que luego irán dentro del mensaje SOAP. Para la invocación en Java de métodos remotos se requiere de stubs, en el caso de los Web Services JAX-WS provee una herramienta, WsImport, capaz de generar dichos stubs de forma automática a partir de los descriptores WSDL. Estos stubs son los que empleara el runtime-system de JAX-WS para convertir las invocaciones java en mensajes SOAP/HTTP.

El siguiente código, a modo de ejemplo, muestra la codificación Java para el envío de la notificación incomingVideoshare, incluida en el EJB que modela el cliente WS para notificaciones de sesión VS.

```
Package com.solaiemes.rcsvs.notifiers;

import com.solaiemes.rcsbox.notifications.webservices.videoshare.stub;
import com.solaiemes.rcsbox.util.WSPortLocator;

@Stateless
public class VideoshareNotifierBean implements VideoshareNotifierLocal {

@Override
public void VideoshareReceived(String wsdlUrl, String username, String
inviteSessionId, String remoteUri) {

VideoshareReceiverWS port = WSPortLocator.getVideosharePort(wsdlUrl);
NotificationResult result = port.incomingVideoshare(username, remoteUri,
inviteSessionId);

}
```

Extracto 3. Codificación para envío de una notificación.

Los métodos de los EJB del API sobre los que se codifica el envío de notificaciones serán accedidos desde el módulo Services, que se encarga de gestionar en última instancia el envío de éstas de acuerdo a los eventos producidos en el Gateway y será además quien proporcione la información que irá en los mensajes SOAP/HTTP .

4.2.3 Módulo Services

El módulo Services codificado en el paquete **com.solaiemes.rcsvs.services**, cuenta con seis EJBs: tres para implementar la lógica de negocio tras cada uno de los Web Services del API (XxxxServiceBean) y otros tres para gestionar el envío de notificaciones a través de los clientes WS del API (XxxxServicesReceiverBean). Cada uno de los EJB de este módulo se ocupa de un WS o un cliente WS en particular.

Los EJBs XxxxServiceBean, atienden llamadas desde los Web Services del API, cuando una acción ha sido invocada, realizan peticiones a los EJB DAO del gestor de datos de acuerdo a los requerimientos de la acción y actuando como clientes de los EJB del módulo SIPServices solicitan a éstos el procesamiento SIP requerido (ver apartado 4.2.3).

En el caso de los XxxxReceiverBean, estos son accedidos desde el módulo SipServices cuando se produce algún evento a nivel SIP o RTP, para que sea notificado a la aplicación usuaria involucrada. Las funciones principales de estos EJB son: conformar los datos que serán incluidos en la notificación, obtener de base de datos la URL para notificaciones de la aplicación usuaria destinataria y solicitar a los clientes del API el envío de la notificación.

VsServiceBean

-Implementa la lógica del servicio web VideoshareWS.
-Comprueba aspectos, como que la aplicación invocante se encuentre registrada en el core, que no haya otra sesión abierta con el mismo usuario cuando se solicite establecimiento de sesión, que realmente haya sesión cuando se solicite cierre, que el usuario RCS-e remoto este en la lista de contactos de la aplicación, etc .
-Solicita al EJB InviteServiceBean del modulo SIPServices, establecimiento, cierre, aceptación o rechazo de sesión VS
-Cuando haya sido invocada la acción sendFileVideoshare solicitara a FileBean el fichero fuente de transmisión

VsReceiverServiceBean

-Sera accedido desde InviteServiceBean cuando se produzca algún evento SIP o RTP durante una sesión VS
-Gestiona el envío de notificaciones de sesión VS a través del cliente VideoShareNotifierBean.
-Cuando haya sido solicitado almacenamiento de video en recepción, realiza la petición correspondiente sobre el EJB FileBean

SessionServiceBean

-Implementa la lógica del servicio web SessionWS.
-Comprueba aspectos como que el usuario invocante ha sido provisionado previamente en el GW.

-Solicitan al EJB RegisterServiceBean, registro y deregistro SIP en el core.
-Controla los temporizadores para refresco de registro de las aplicaciones.
-Accede a ContactsServiceBean para incializar y parar el mecanismo de polling de OPTIONS

SessionReceiverServiceBean

-Será accedido desde RegisterServiceBean, cuando algún evento SIP durante el registro o deregistro de la aplicación se produzca.
-Gestiona el envío de notificaciones resultado del inicio o finalización de sesión de la aplicación en el core a través del cliente SessionNotifierBean.

ContactsServiceBean

-Implementa la lógica del servicio web SessionWS.
-Actualiza la lista de contactos-usuarios (CONTACT) de las aplicaciones, añade o borra contactos cuando la acción correspondiente ha sido invocada.
- Controla el temporizador del mecanismo de polling de OPTIONS de cada aplicación
-Solicita al EJB OptionServiceBean, envio de mensajes SIP OPTION para cada contacto de la aplicación.

ContactsReceiverBean

-Será accedido desde el EJB OptionServiceBean, cuando un evento SIP durante un intercambio de mensajes SIP OPTION se produzca.
-Actualiza el estado de los contactos de las aplicaciones (CONTACTS) , en cuanto a la disponibilidad y capacidad de servicio VS de éstos.
-Gestiona el envío de notificaciones acerca de los cambios producidos en la lista de contactos de una aplicación, a través del cliente ContactsNotifierBean

Tabla 6. EJBs del módulo Services

4.2.4 Módulo SipServices

El módulo SipServices gestiona la comunicación del Gateway con los usuarios RCS-e mediante el protocolo SIP. Esta comunicación es llevada a cabo mediante sesiones donde se produce un intercambio de mensajes y respuestas SIP entre el Gateway y el usuario. En el Gateway puede haber tres tipos de sesiones SIP:

-Sesiones VS, iniciadas por el envío o recepción de un mensaje INVITE.
-Sesiones generales en el core, mediante mensaje REGISTER.
-Sesiones para el intercambio de la capacidad VS, mediante OPTION.

SipService implementado en el paquete software **com.solaiemes.rcsvs.sip.services,** cuenta con tres EJB de sesión, cada uno para la gestión de un tipo de sesión. Estos EJB son sin estado, por lo que empleará los EJB DAO del gestor de datos para mantener el estado de las sesiones, representadas como entradas en base de datos (INVITE_INFO, REGISTER_INFO, OPTION_INFO).

Adicionalmente a la sesiones SIP también gestiona las sesiones de video RTP creadas en el servidor de video, manteniendo el estado de las mismas a través del bean VsSessionInfoDaoBean, encargado de las entradas de la tabla VS_SESSION_INFO.

InviteServiceBean
-Mantiene el estado del establecimiento VS a través de InviteInfoDaoBean. -Recibe peticiones de VsServiceBean y notifica eventos a VsReceiverServiceBean -Gestiona el envío y recepción de los mensajes SIP INVITE y BYE así como el de las respuesta 200Ok, ACK o de error a través de los componentes de la pila SIP InviteActionBean y InviteServlet, respectivamente. -Accede a las funciones del SDPNegotiationBean, para obtener los SDP que se incluirán en mensajes INVITE y respuestas 200Ok, durante el establecimiento VS. -Crea, gestiona, activa y termina sesiones VS, para las transmisiones/recepciones de video RTP que se llevan a cabo en el servidor de video a través del componente VsActionBean. Mantiene el estado de estas sesiones de video
RegisterServiceBean
-Mantiene el estado de registro de las aplicaciones en el core a través de RegisterInfoDaoBean. -Recibe peticiones de SessionServiceBean y notifica eventos a SessionReceiverServiceBean -Gestiona el envío de mensajes SIP REGISTER así como el de las respuesta 200Ok o de error, a través de los componentes de la pila SIP RegisterActionBean y RegisterServlet respectivamente.
OptionServiceBean
-Mantiene el estado del intercambio de la capacidad VS de cada uno de los contactos-usuarios de las aplicaciones a través de EJB OptionInfoDaoBean. -Recibe peticiones de ContactsServiceBean y notifica eventos a ContactReceiverServiceBean -Gestiona el envío de mensajes SIP OPTION así como el de las respuesta 200Ok o de error a través de los componentes de la pila SIP OptionActionBean y OptionServlet, respectivamente.

Tabla 7. EJBs del módulo SipServices

Todos estos EJBs accederán, durante la sesión SIP correspondiente, a la tabla de base de datos SIP_USERS, a través del EJB SipUserDaoBean, para obtener la información SIP con la que fue aprovisionada la aplicación. En el caso de InviteServiceBean ademas tambien accederá mediante MediaUserDaoBean a la información de medias de la aplicación, que empleara durante el negociado que se realiza en el establecimiento VS.

4.2.5 Pila SIP

El Gateway de cara al core IMS/RCS-e funcionara para cada una de las aplicaciones como UAC (User Agent Client) y UAS (User Agent Server), recibiendo y enviando mensajes SIP al mismo tiempo. Debido a esto, la pila ha sido modelada en dos paquetes, uno contendrá los EJB sobre los que se implementa el cliente SIP y otro

donde se empleará SIPServlet para recibir peticiones SIP realizando funciones de servidor.

-Agente Cliente (com.solaiemes.rcsvs.sip.action)

Siguiendo con la distribución del módulo SIPService, las funciones para la generación y el envió de mensajes SIP se ha separado en tres EJBs. Cada uno de los métodos de estos EJB es dedicado en exclusiva a la composición y envío de un tipo de mensaje SIP. Los métodos serán invocados por los EJB del módulo SIPServices y serán éstos los que les proporcionen a través de los parámetros del método la información que irá en las cabeceras de los mensajes SIP.

InviteActionBean
-Generación y envío de mensajes SIP relacionados con el establecimiento de sesión VS -Métodos: sendInvite(), sendAck(), sendError(), sendOk(), sendCancel() y sendBye().
RegisterActionBean
-Generación y envío de mensajes SIP relacionados con el registro de la aplicación en el core IMS/RCS-e -Métodos: sendRegister(), sendUnregister(), sendRegisterAuth()
OptionActionBean
-Generación y envío de mensajes SIP relacionados con el intercambio de capacidad VS. -Métodos: sendOption(), send200Ok(), sendError()

Tabla 8. EJBs del cliente SIP

-Agente Servidor (com.solaiemes.rcsvs.sip.servlets)

La parte servidora de la pila ésta formada por tres SipServlets, cada uno se encarga de recibir y procesar las peticiones y respuestas de los mensajes SIP de un tipo de sesión (ver apartado 4.2.3). Cada uno de los métodos de estos Servlets es dedicado a una petición o respuesta en particular. Primero realizan un procesado consistente en el parseo del contenido del mensaje SIP y luego acceden a los EJBs del módulo SIPServices, donde se harán las correspondiente actualizaciones de la sesión y todo el procesamiento que esta implique.

InviteServlet
-Recepción y procesado de mensajes SIP relacionados con el establecimiento de sesión VS -Métodos: doInvite(), doAck(), doErrorResponse(), doSuccessResponse(), doCancel() y doBye().

RegisterServlet
-Recepción y procesado de mensajes SIP relacionados con el registro de la aplicación en el core IMS/RCS-e -Métodos: doErrorResponse(), doSuccessResponse()
OptionServlet
-Recepción y procesado de mensajes SIP relacionados con el intercambio de capacidad VS. -Métodos: doOption(), doSuccessResponse(), doErrorResponse()

Tabla 9. SipServlets del servidor SIP

Para la implementación de la pila SIP se ha empleado el API de Java Sip Servlet (javax.servlet.sip). Empleando este API, cada sesión SIP es modelado como un objeto *SipApplicationSession* que cuenta con un identificador interno mediante el cual se mantiene el estado de la sesion (el intercambio de mensajes SIP es asincrono). Este objeto contendrá una o varias peticion SIP (mensajes SIP) como instancias de la clase *SipServletRequest*. Esta clase cuenta con todos los métodos necesarios para la generación y parseo de la información contenida en el menaje SIP, así como un método *send*() para llevar acabo el envío. Las posibles respuesta a cada petición son instancias de la clase *SipServletResponse* y son generadas por la propia petición con el método *createResponse()*. Las respuestas como es lógico formarán parte de la misma sesión SIP que la petición que las ha originado [56].

En las siguiente extracto se presenta el código del metodo senInvite(), donde se puede observar como empleando la interfaz *SipFactory* ofrecida por el API Sip Servlet, se crea una sesión SIP y dentro de ésta una petición INVITE a la que se le van añadiendo las cabeceras para luego ser enviada.

```
public void sendInvite(SipUser user, InviteInfo inviteInfo, SipBodyContent
sipBodyContent) {

SipApplicationSession sas = sipFactory.createApplicationSession();

String from =
sipFactory.createSipURI(user.getSipUser(),user.getSipDomain()).toString();
String to = inviteInfo.getRemoteUri();

SipServletRequest req=sipFactory.createRequest(sas, "INVITE", from, to);

      req.addHeader("User-Agent", user.getSipUserAgent());
      req.addHeader("Expires", Integer.toString(user.getSipExpires()));
      req.addHeader("Supported", "timer");
      req.addHeader("Session-Expires", "3600;refresher=uac");
      req.addHeader("Contribution-ID", "1234045646543");
      req.setHeader("P-Preferred-Identity", from);

      Parameterable contact = req.getParameterableHeader("Contact");
      contact.setParameter("+g.3gpp.cs-voice", "");
      req.addHeader("Accept-Contact", "*;+g.3gpp.cs-voice");
```

```
	req.addSipBodyContent(req, sipBodyContent);

	inviteInfo.setSessionId(sas.getId());

	req.send();
}
```

Extracto 4. Código para envío de mensaje SIP INVITE.

Para el caso de los SipServlet hay que implementar cada uno de los métodos doXxx() a través de los que se procesa las peticiones o respuesta recibidas. En nuestro caso, siempre que se reciba una petición o respuesta, obtendremos el identificador de la sesión SIP (SipApplicationSession) a la que pertenece y lo pasaremos al EJB del módulo SIPService correspondiente, que se encargará de todo el procesado. Ante la llegada de una petición, la respuesta será generada directamente en el método doXxxx (), si no se requiere de la decisión de la aplicación usuaria (por ejemplo cuando se acepta o rechaza una petición de INVITE).

En el siguiente extracto se pone como ejemplo la codificación del método doBye() dentro de InviteServlet. Se puede observar que en este caso, directamente se crea y se manda la respuesta 200Ok para luego invocar al método byeReceived() del EJB InviteServiceBean, que se encarga de la actualizaciones de la sesión y otras gestiones que implica la petición recibida. En este caso se cerrará la sesión RTP en curso, se borrara la sesión de la tabla INVITE_INFO y se le indicará a VideoShareNotifierBean que se lo notifique a la aplicación involucrada.

```
protected void doBye(SipServletRequest req) {

	req.createResponse(SipServletResponse.SC_OK).send();
	inviteService.byeReceived(req.getSession().getId());

}
```

Extracto 5. Código para recepción de mensaje SIP BYE.

Los Servlets serán desplegados en el contenedor de Sip Servlets del servidor Mobicents, para lo cual será necesario incluir en éste el fichero de despliegue sip.xml, donde se le indica los puntos de escucha de las peticiones SIP, de modo que el servidor pueda localizar los métodos a invocar para cada tipo de petición SIP recibida. Aquí se presenta el extracto del fichero sip.xml para el servlet InviteServlet, para el resto el uso de las etiquetas XML es similar.

```
<sip-app>
	<app-name>RcsBox</app-name>
	<display-name>RcsBox</display-name>
	<description>RcsBox</description>
</servlet-mapping>
```

```
      <servlet-mapping>
      <servlet-name>Invite</servlet-name>
      <pattern>
            <or>
                  <equal>
                        <var>request.method</var>
                        <value>INVITE</value>
                  </equal>
                  <equal>
                        <var>request.method</var>
                        <value>BYE</value>
                  </equal>
                  <equal>
                        <var>request.method</var>
                        <value>CANCEL</value>
                  </equal>
                        </or>
                  </pattern>
            </servlet-mapping>
</servlets>
      <servlet>
            <servlet-name>Invite</servlet-name>
            <display-name>Invite</display-name>
            <description>Invite</description>
            <servlet-class>
                  com.solaiemes.rcsbox.servlet.Invite
            </servlet-class>
            <load-on-startup>5</load-on-startup>
      </servlet>
```

Extracto 6. Configuración para despliegue de SIPServlets

4.2.6 Acceso a repositorio

El módulo de acceso a repositorio se correponde con el paquete software **com.solaiemes.rcsvs.files y** está constituido por un sólo EJB sobre el que se ha modelado un cliente HTTP para la comunicación con el repositorio de videos, que actúara como servidor HTTP. En el EJB se ha implementado dos métodos, uno para la obtención de un fichero del repositorio y otro para almacenar fichero en repositorio. Estos métodos son invocados por VideoShareService y VideoShareReceiverService, respectivamente, cuando se requiera fichero como fuente de transmisión y cuando la aplicación haya solicitado almacenamiento de fichero en recepción.

FileBean
-getFileToRepository(): Abre una conexión HTTP con el repositorio donde está localizado el fichero (url del fichero) y manda una petición HTTP GET. El fichero será transmitido como un flujo de byte por la conexión abierta, desde el repositorio hasta el Gateway.
-putFileToRepository(): Abre una conexión HTTP con el repositorio de video y genera una petición

POST, incluyendo en el cuerpo del mensaje: el fichero en si como un array de bytes, el identificador del espacio de almacenamiento donde guardarlo y el nombre del fichero con el que se desea almacenar. En la respuesta el repositorio enviara la URL HTTP donde almacenará el fichero.

Tabla 10. Métodos del EJB del módulo de acceso a repositorio.

Para la implementación de estos métodos se ha empleado, la clase *HttpURLConnection* de la librería java.net para el establecimiento de conexiones HTTP y clases de la librería java.io tales como *InputStream* y *BufferedInputStream*, que nos permitirán recibir y el manejar los ficheros como flujos de bytes [40] [62].

4.2.7 Acceso a servidor de video.

Este módulo implementado en el paquete **com.solaiemes.rcsvs.vsActions**, contiene un único EJB, VsActionsBean, sobre el que se establece la comunicación con el servidor de video para el control de las sesiones de video RTP. El mecanismo empleado para tal fin es el de invocación remota de métodos RMI.

El servidor de video cuenta con una interfaz de acceso (RemoteMedialib.java), a través de la cual se pueden crear, activar y desactivar sesiones RTP, pasándole los parámetros requeridos. En el EJB VsActionsBean se creará un objeto remoto de dicha interfaz a través del que se harán las distintas invocaciones. Este EJB cuenta con una serie de métodos que encapsulan las llamadas al servidor de video. Estos métodos a su vez serán accedidos desde InviteServiceBean que controla el establecimiento de sesiones VS y decide cuando crear y en qué condiciones las sesiones RTP, cuando activarlas y cuando pararlas.

En el siguiente extracto se presenta como se lleva a cabo la invocación remota para la activación de una sesión RTP. De forma similar se hará para la creación de la sesión y para la terminación de la sesión.

```
public void activateSession(VsSessionInfo vsSessionInfo) {

remoteMedialibAddress =ConfigValuesDao.getConfigValue(REMOTEMEDIALIB_ADDRESS)
RemoteMedialib  mediaRMI =(RemoteMedialibRmi)Naming.lookup(rtpStackAddress);
mediaRMI.activateSession(vsSessionInfo.getId());

}
```

Extracto7. Invocación remota para la activación de sesión RTP en el servidor de video

Del mismo modo, también se dará comunicación en el otro sentido. Desde la interfaz del servidor se notificarán eventos que se producen durante la sesión RTP: error durante la misma, final de flujo RTP y finalización de grabado en fichero. En este caso el servidor de video realiza invocación remota sobre métodos alojados en VsActionBean para tal fin.

VsActionBean
-createSession(): para crear una sesión RTP. Por parámetros se pasarán todos los datos requeridos: tipo de sesión (envío de fichero, envió en vivo o recepción) puertos de la conexión, fuente, y parámetros negociados para la transmisión. -activateSession(): activación de la sesión previamente creada. -terminateSession(): finalización de sesión previamente creada. -receivedError(): evento producido por un error durante la transmisión -receivedEOS(): evento producido cuando se detecta final de flujo RTP -receivedFileRecorded(): cuando finaliza una sesión en recepción. A través de esta notificación se pasa el fichero grabado, para su posterior almacenamiento en repositorio

Tabla 11. Métodos del EJB del módulo de acceso a servidor de video

4.2.8 Gestor de medias.

El módulo Gestor de medias es el encargado de la negociación de medias, codecs y demás parámetros de transmisión. Este módulo se ha codificado en el paquete **com.solaiemes.rcsVS.sdp** en el que se incluye un sólo EJB que cuenta con dos métodos, uno para la intersección de SDPs y otro para la comprobación de SDPs:

El primero será aplicado cuando el originante de la sesión VS sea el usuario RCS-e remoto. Al recibir un mensajes SIP INVITE, el EJB InviteSipService invocará a este método, pasándole por parámetros el SDP recibido en el INVITE y el SDPtemplate de la aplicación involucrada, el cual fue aprovisionado en la tabla MEDIA_USERS, cuando se dio de alta a la aplicación. La intersección entre ambos se hará de acuerdo al criterio definido en el requisito 8 (ver apartado 3.2.1). El resultado de la intersección será un nuevo SDP, que será devuelto a InviteSipService para que lo incluya en la respuesta 200Ok del establecimiento.

El método de comprobación se aplicará cuando el Gateway es el originante. En este caso el SDP incluido en el mensaje INVITE se corresponde con el SDPtemplate de la aplicación. El método será invocado por InviteServiceBean cuando se reciba el 200Ok procedente del usuario RCS-e. En este caso solamente se hace un chequeo del SDP para comprobar que es compatible con la aplicación, de acuerdo al SDPtemplate de la misma, si es así, se indica a InviteServiceBean que confirme con ACK, y si no lo es, se le indica que mande el mensaje SIP de error correspondiente.

Este EJB además contiene un conjunto de métodos get(), mediante los que proporcionará los parámetros de transmisión contenidos en el SDP intersectado o testeado, de forma individualizada, tras el correspondiente parseo. InviteSipServlet tomará estos parámetros para la creación del la sesión de video, pasandolos al servidor de video para que este transmita/reciba de acuerdo a ellos.

Adicionalmente a los métodos para negociado de medias, este EJB implementa el mecanismo de gestión de puertos para transmisión RTP, descrito en el capitulo anterior (ver apartado 3.3.2). Para mantener el estado de puertos ocupados utilizaremos un

Mapa de Java, donde se pueden guardar pares clave-valor, en nuestro caso la clave será el identificador de la sesion INVITE y como valor se tomará el número de puerto empleado por la sesión INVITE. De esta forma asignaremos y liberaremos puertos por petición del EJB InviteSipSerlet, a traves del identificador de sesión INVITE proporcionado por éste. InviteSipServlet cuando inicie establecimiento pedirá puertos para incluirlos en el SDPtemplate de la aplicación, y cuando termine la sesión VS antes de eliminar el registro correspondiente de la tabla INVITE_INFO, lo notificará a este EJB para que los libere.

SDPNegotiationBean
-IntersecSDP(): intersección del SDP recibido en INVITE y el SDPtemplate de la aplicación. -ValidateSDP(): validación del SDP recibido en respuesta 200Ok -getBitrate(), getPortDest(), getHostDest(), getCodec(), getPayLoad(), getFrameRate(), getWidth(), getHigh(). -Map<String, Integer> managerPort -getFreePort(): asignación de puerto libre, se añade a managerPor como ocupado. -releasePort(): se libera puerto, se borra de managerPort.

Tabla 12. Métodos del EJB del módulo gestor de medias

Para la implementación de los métodos que gestionan SDPs, se ha empleado como base la librería javax.sdp. Esta librería implementa la especificación JAIN-SDP [63], donde se define la implementación del protocolo SDP en java, como complemento a la especificación JAIN-SIP [64]. JAIN-SIP es la implementación de referencia de señalización SIP en java, y es la base sobre la que se ha desarrollado el API de SIP Servlets, por lo que javax.sdp será totalmente compatible con nuestra pila SIP.

Mediante esta librería los SDP son manejados como objetos de la clase *SessionDescription*, los cuales a su vez contienen objetos *MediaDescription* que representan los distintos medias (m=) y sus atributos (a=), permitiendo asi un acceso totalmente programado a los campos que conforman los SDPs, los medias y los parámetros de estos medias. Además mediante la interfaz *SDPFactory* proporcionada por esta librería se puede codificar y decodificar SessionDescription de/a Strings, lo que nos permitirá gestionar los SDPtemplate de las aplicaciones aprovisionados en base de datos y realizar fácilmente parseos para obtener los parámetros de transmisión negociados.

4.2.9 Diagrama de componentes del servidor de control.

En el siguiente diagrama se presenta el conjunto de todos los componentes vistos, sobre los que se modela las distintas funcionalidades del servidor de control y señalización (los componentes del sistema gestor de datos no se han incluido para evitar una excesiva complejidad del diagrama). Las dependencias entre componentes indicada

por las flechas discontinuas, nos da una idea de cómo es procesado el flujo de información a través de los distintos módulos:

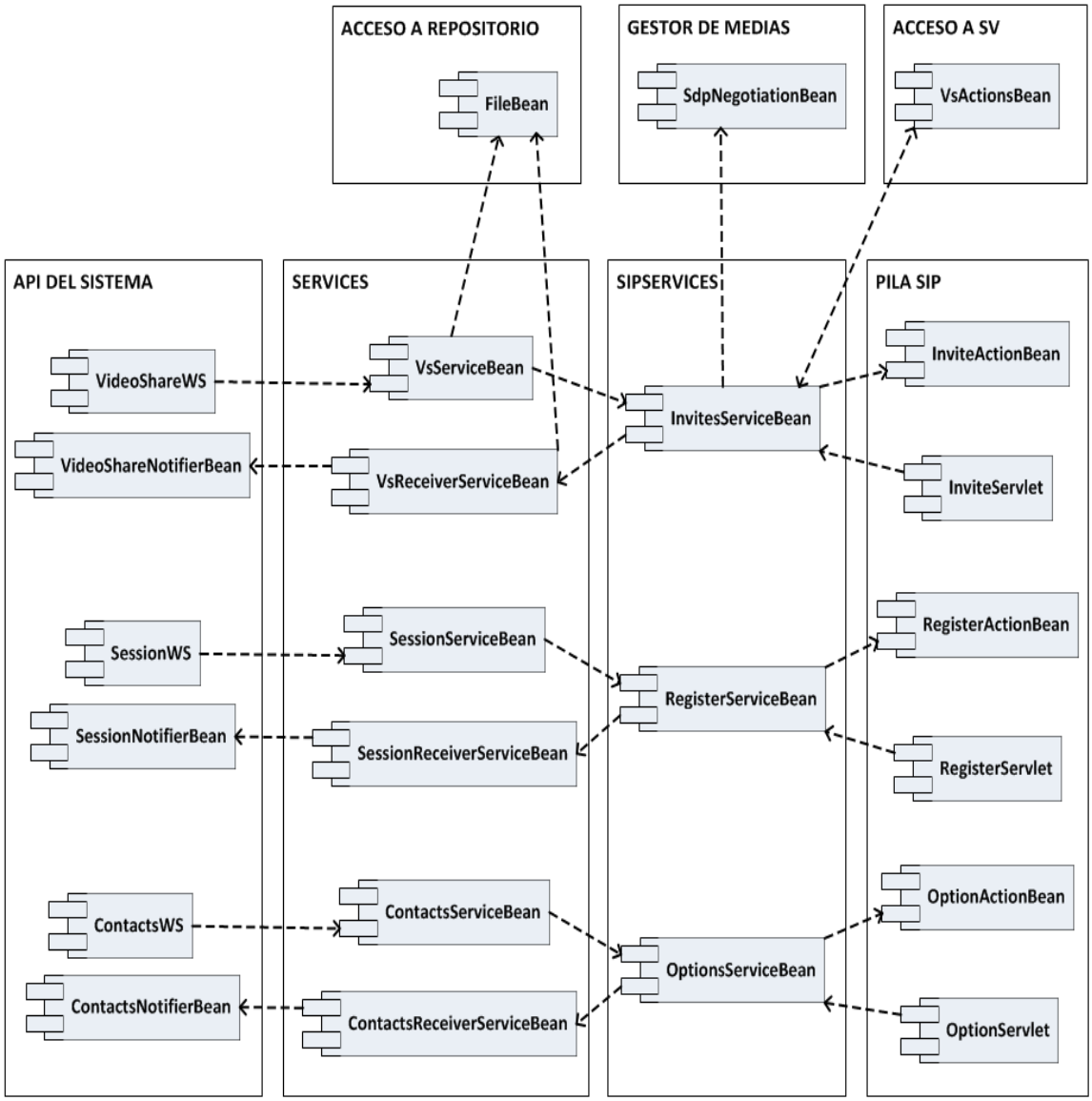


Figura 34. Diagrama de componentes del servidor de control y señalización.

4.3 Servidor de video.

El servidor de vídeo se ha implementado como un módulo aparte, pudiéndose ejecutar en una máquina distinta al servidor de control y señalización, con la intención de hacer reparto de cargas y facilitar la escalabilidad del sistema en fases futuras del proyecto.

La implementación del servidor se ha realizado empleando el framework de Gstreamer. Nos basaremos en su arquitectura de pipelines para procesado de media y se accederá su API para implementación de aplicaciones mediante Gstreamer-Java. Aunque la implementación de este módulo también se ha hecho enteramente en Java, en este caso no ha sido necesario usar las APIs de la plataforma JEE, por lo que tampoco se hace necesario emplear un servidor de aplicaciones JEE como es el caso de JBoss para el servidor de control y señalización, en este caso, el servidor de video sólo requerirá de la máquina virtual de Java para poder ser ejecutado. El hecho de que haya sido también implementado en Java permitirá que ambos servidores puedan comunicarse mediante RMI.

4.3.1 Pipelines para procesamiento de media

En el apartado 3.3.3, ya se habló del funcionamiento del servidor de video, basado en una arquitectura de pipelines donde se llevaba a cabo un procesamiento sobre el vídeo procedente de una fuente, tal como un fichero o flujo RTSP/RTMP, para luego ser enviado como flujo RTP por una conexión UDP, en el caso de transmisión. En el caso de recepción, el procesamiento se realizaba sobre el video contenido en el flujo RTP procedente de un terminal móvil, el cual terminaría por ser guardado en un fichero y/o entregado a servidor Liveserve para su publicación mediante RTSP/RTMP.

Como ya vimos, el servidor de control y señalización contempla tres tipos de sesión de video: envío de video pregrabado, envío de video en vivo y recepción de video. Cada una de estas sesiones requería de un procesamiento distinto, dividido en etapas o funciones de procesado a lo largo de un pipeline. La funcionalidad requerida para ello ha sido tomada de Gstreamer que provee a través de sus diferentes plugins un conjunto de elementos que realizan funciones especificas de procesamiento (ver apartado 4.1.3). A través del API de Gstreamer podremos crear los elementos requeridos y enlazarlos en el orden apropiado, configurando así pipelines para cada tipo de sesión.

A continuación se presentan el esquema de cada uno de los pipelines de elementos empleados para cada tipo de sesión. En la posterior tabla se resume la función de cada elemento así como las propiedades configurables que poseen éstos, y que han sido tenidas en cuenta de acuerdo a los parámetros de transmisión/recepción que son negociados durante el establecimiento de una sesión VS (si no son negociados, se pondrán por defecto, según el perfil de medias de la aplicación).

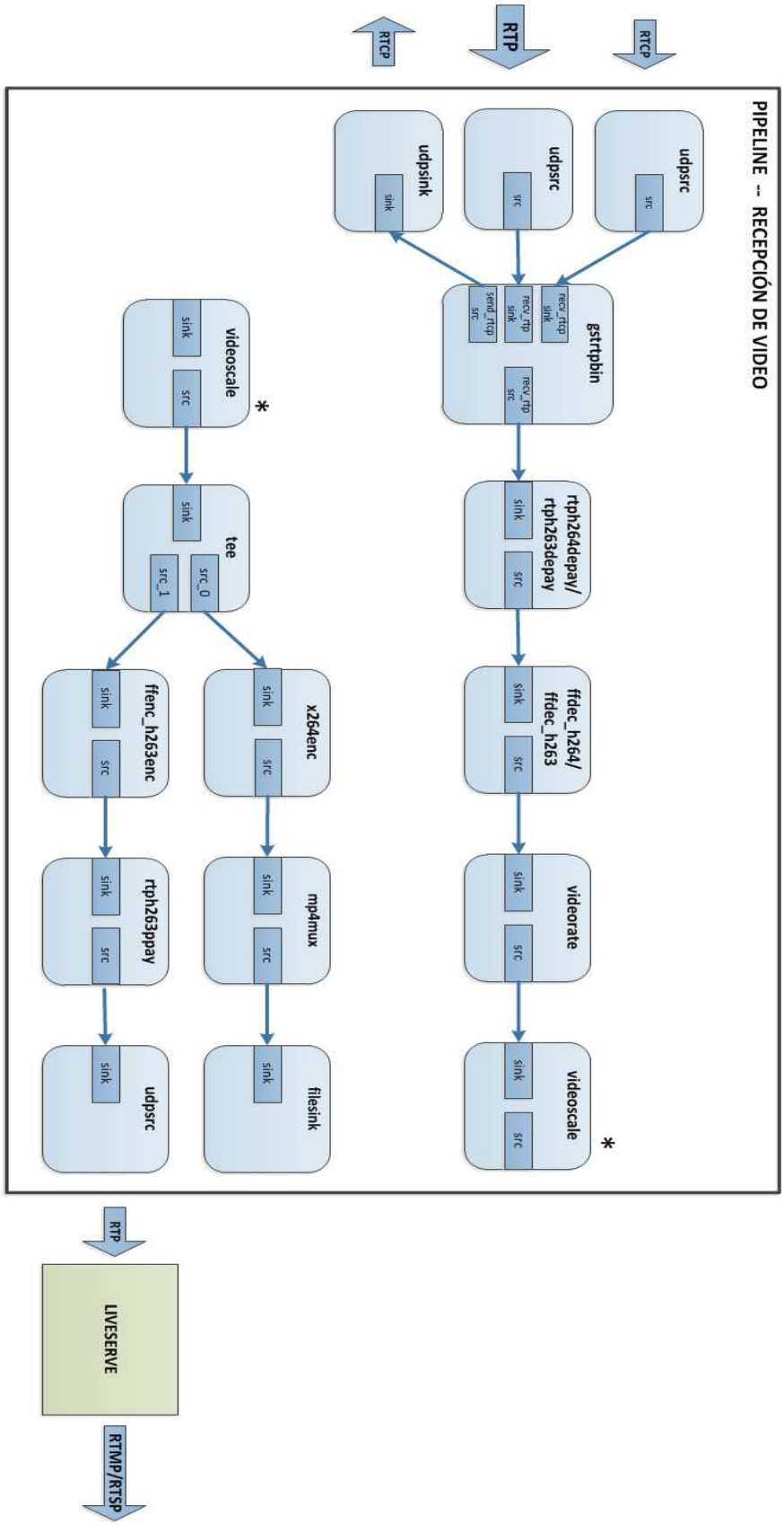


Figura 35. Pipeline para recepción de vídeo

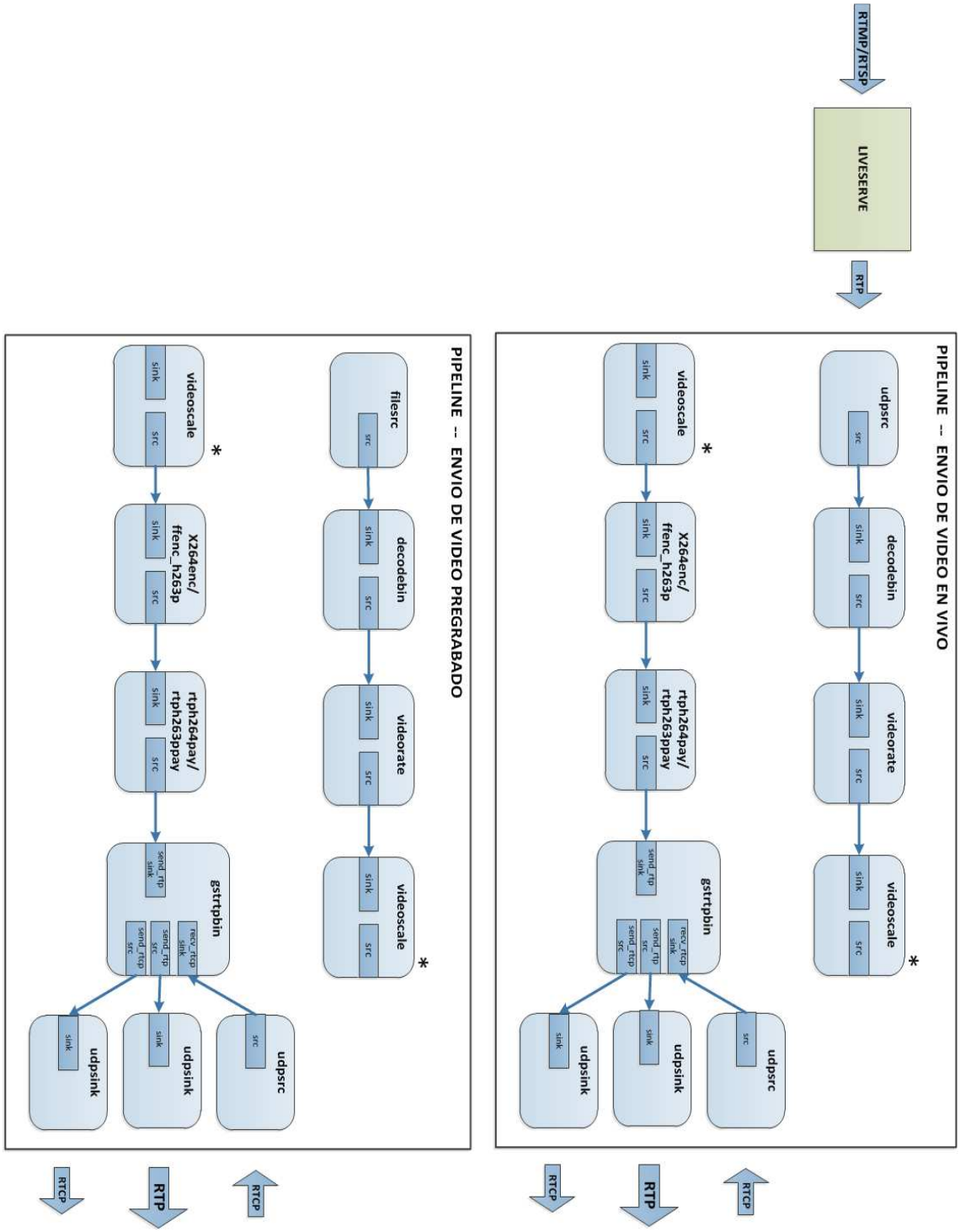


Figura 36.Pipelines para envío de video

udpsrc -- udpsink

-Elementos para el envío y recepción de datos a traves de la red via UDP. Los protocolo RTP y RTCP esta implementado sobre UDP, por lo que se necesitara por cada pipeline para sesion de video, uno de estos elementos para RTP (enviado o recbido) y dos para RTCP, uno para paquete SR (Send Report) y otro para los RR (Receiver Report).

Propiedades configurables:

- host y port donde se recibirá los datos en el caso de udpsink.
- host y port a donde se enviarán los datos en el caso de udpsrc.
- caps: para filtrado de tipo de datos a recibir o enviar. En nuestro caso será puesto como "application/x-rtp,media=video,encoding-name=H264/H263", indicando el tipo de datos, el tipo de media y el tipo de codificación.

gstrtpbin

-Este elemento es un bin proporcionado por Gstreamer que contiene toda la funcionalidad requerida para gestionar el envío y recepción de paquetes RTP. Este elemento tambien gestiona el control de sesión mediante generación, envío y recepcion de paquetes RTCP.

rtph264pay / rtph263pepay -- rtph263pdpay / rtph264pay

-Estos elementos son empleados para el paquetizado y depaquetizado RTP del video. Dependiendo del codec negociado durante el establecimiento se emplearan las versiones para h264 o h263 plus.

Propiedades configurables:

- Payload de los paquetes rtp

x264enc / ffenc_h263p -- ffdec_h264 / ffdec_h263p

-Estos son los elementos codificadores y decodificadores de video. Se empleará las versiones para h264 y h263 plus para el transcodificado del video según la negociación realizada durante el establecimiento de sesión y los requerimiento en recepción (almacenamiento en h264 y publicacion en h263 plus).

Propiedades configurables:

- Bitrate empleado en la codificación. Se pondrá en emisión de acuerdo a lo negociado con el terminal móvil durante el establecimiento. En recepción es adaptativo.

videorate

-Este elemento tira, duplica y ajusta las tramas de video a partir del timestamp de estos. Empleandolo junto con un filtro nos permitira ajustar el framerate del flujo según lo negociado en el establecimiento.

videoscale

-Este elemento cambia el tamaño del los frames de video. Empleando con un filtro de podremos cambiar la resolución del video según lo negociado durante el establecimiento (CIF, QCIF, etc).

mp4mux

-Este elemento es un mezclador de medias para MPEG-4 (ficheros .mp4). Aunque en nuestro caso solo tenemos video, es necesario emplearlo para el correcto grabado en fichero.

decodebin
- Se trata de un elemento “inteligente” provisto por Gstreamer para convertir cualquier media, independientemente de su formato y codificación, a datos en crudo “raw”, para su posterior procesado. Este elemento en sí es un bin de elementos que realiza entre otras funciones de demultiplexado, decodificado y depaquetizado. Solo se ha empleado en envío porque se detecto que en recepcion enlazado con gstrtpbin no funciona bien, por ello que para ese caso se emplee decodificador y depaquetizador.
tee
-Elemento para la división del flujo de datos. En nuestro caso lo emplearemos en recepción cuando se haya solicitado almacenamiento en fichero y publicación RTMP/RTSP al mismo tiempo.
filesink – filesrc
-Estos elementos se emplean para la escritura y lectura, respectivamente, de datos de/a fichero. Se emplearán para obtener el video de un fichero, en el caso de envío de video pregrabado y para grabar video en fichero en el caso de recepción. Propiedades configurables: • Localización del fichero (directorio local), donde se realizara la lectura o escritura de datos.

Tabla 13 .Elementos de los pipelines.

- **Bus de eventos.**

Durante el funcionamiento de un pipeline, es decir mientras lleva a cabo el procesamiento del flujo de media, pueden ocurrir diferentes eventos, tales como un fallo motivado por una mala configuración de elementos, datos defectuosos, la llegada de datos a través de una conexión, la activación de un elemento del pipeline, etc. Los elementos de Gstreamer tienen la capacidad para detectar internamente este tipo de eventos, los cuales serán comunicados a la aplicación a través de un mecanismo provisto por Gstreamer llamado Bus.

Cada pipeline creado con el API de Gstreamer lleva asociado un Bus de eventos. Es parte de la aplicación indicar al Bus que tipo de eventos quiere que le sean comunicados. En nuestro caso en particular, emplearemos el Bus para saber cuando se produce un error durante la transmisión o recepción de video (evento Bus.error) y para saber cuando la transmisión o recepción de video ha terminado (evento Bus.eos).

4.3.2 Modelado software del servidor de video

En el siguiente diagrama se presenta las clases con las que se ha modelado el servidor de video:

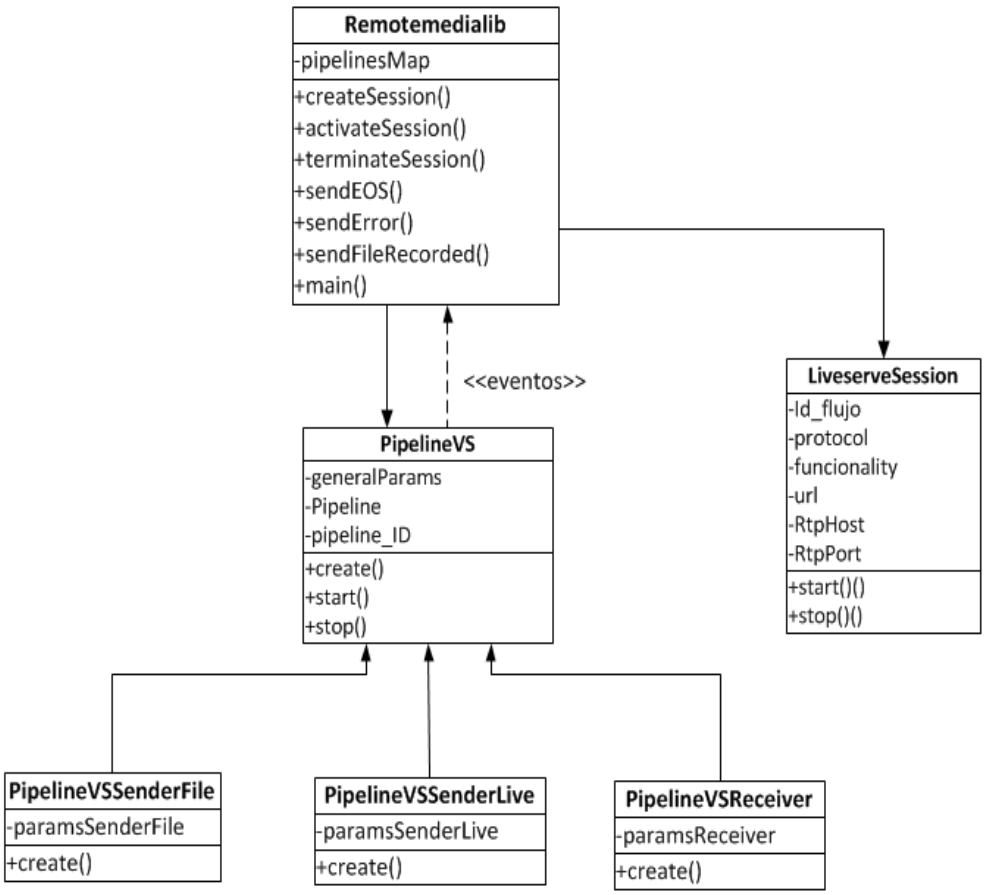


Figura 37. Diagrama de clases del servidor de video.

- **Modelado de Pipelines**

PipelineVS.java es una clase abstracta que modela un pipeline de sesión VS genérico, mientras que las clases PipelineVSXxx.java son hijas de la primera y representan cada uno de los tipos de pipelines que contempla el servidor de video.

-El atributo pipeline común a todas ellas, es un objeto del API de Gstreamer para gestionar un pipeline de procesamiento en su conjunto.

-**Los método start() y stop(),** mediante los que se inicia y para el procesado de media, también son comunes, por lo que serán implementados en la clase padre. La implementación de estos métodos consiste sólo en una llamada a las funciones del API de Gstreamer para parar y arrancar un pipeline: pipeline.stop() y pipeline.start()

-**El método create()**, contiene todas las llamadas al API de Gstreamer para la creación de los elementos del pipeline, la configuración de éstos a través de sus propiedades, y el enlazado de los mismos. También será necesario añadirlos al pipeline para que pueda formar parte del conjunto. En el siguiente extracto de código se puede observar cómo se llevan a cabo estos pasos para dos elementos en particular, del mismo modo se hará para todos los elementos que constituyan el pipeline.

```
    Element encoder = ElementFactory.make("x264enc", null);
    Element payloader = ElementFactory.make("rtph264pay", null);

    encoder.set("bitrate", paramsSenderFile.getBitrate());

    payloader.set("pt", paramsSenderFile.getPayloadType());

    Element.linkMany(encoder, payloader);

    pipeline.addMany(encoder,payloader);
```

Extracto 8. Creación, configuración y enlazado de elementos Gstreamer.

En este método también haremos la inscripción de los eventos de error y final de stream sobre el Bus del pipeline, indicándole los métodos a invocar cuando se produzca alguno de estos eventos. En este caso pararemos el procesado y luego lo notificaremos al servidor de control y señalización a través de la interfaz del servidor de Video.

```
Bus bus = pipeline.getBus();

bus.connect(new Bus.ERROR() {
public void errorMessage(GstObject source, int code, String message) {
       pipeline.stop()
       sendError(pipeline_ID);
}});

bus.connect(new Bus.EOS() {
public void endOfStream(GstObject source) {
      pipeline.stop();
      sendEOS(pipeline_ID);
}});
```

Extracto 9. Incripción de eventos en el Bus de Gstreamer.

Cada tipo de pipeline estará constituido por diferentes elementos, por lo que la implementación de este método será particular para cada una de las clases PipelineVSXxxx.java.

-Los parámetros para la configuración de los pipelines (puertos RTP, códecs, bitrate, etc.). constituyen el resto de atributos de estas clases. Algunos serán comunes y otros propios de cada tipo de Pipeline.

- **Remotemedialib**

Esta clase constistuye el interfaz de acceso al servidor de video para el servidor de control y señalización. El EJB VsActionBean, mediante un objeto remoto RMI de esta clase, invocará los métodos createSession(), activateSession() y terminateSession(), durante el establecimiento, una vez establecida y cuando finalice una sesión VS en el servidor de control y señalización

-createSession(): El tipo de sesión VS, el identificador de sesión VS, y los parámetros de transmisión negociados son pasados como parámetros de esta función cuando es invocada. A partir del tipo de sesión, se creará una instancia de la clase PipelineVSXxx.java correspondiente, asignándole a sus atributos valores de acuerdo a los parámetros de transmisión recibidos. Después sobre esta instancia se invocará el método create(), generándose así el pipeline correspondiente. Esta instancia que representa el pipeline asociado a la sesión VS será almacenada en un Mapa Java (PipelineMap), empleando como clave el identificador de la sesión VS, de forma que quedará asociada a dicha sesión VS y pudiendose ser recuperado en el futuro cuando se requiera activar o desactivar el pipeline.

-activateSession(): Recibiremos por parámetro el identificador de sesión a activar, a partir del cual podemos recuperar la instancia PipelineVS asociada a dicha sesión, previamente creada y almacenada en el mapa PipelineMap. Luego sobre esta instancia invocamos el método start(), de forma que comenzará el procesado y la transmisión/recepción de video.

- terminateSession(): procederemos del mismo modo que el caso anterior para obtener el pipeline sobre el cual, en este caso, se invocará el método stop() para finalizar la transmisión/recepción de video. Después se elminará la instancia del mapa PipelineMaps.

-main(): este es el punto de entrada al servidor de video cuando es ejecutado y constituye un requerimiento del mecanismo RMI de comunicación. Aquí es donde el objeto remoto de la clase Remotemedialib es "publicado" para que pueda ser accedido desde VsActionBean .

```
RemoteMedialib remoteMedialib = new RemoteMedialib();
Naming.rebind(conf.getIpMediaServer() + "/RemoteMedialib",remoteMedialib);
```

Extracto 10.Publicado del objeto remoto Remotemedialib

La clase Remotemedialib tambien establece comunicación con el servidor de control, en sentido contrario para comunicarle eventos que se producen durante el procesado del video y para pasarle los ficheros de video grabados en local. Las notificaciones se comunicarán también mediante invocación remota de métodos, que en este caso pertenecen al EJB VsActionBean como ya vimos. En este caso para obtener una instancia del EJB sobre la que se pueda realizar la invocación se emplea un mecanismo provisto por Java llamado ServiceLocator, el cual emplea las librerías JNDI de JEE para localizar un componente dentro de un servidor de aplicaciones JEE.

-sendEOS() y sendError(): Estos métodos son invocados cuando el Bus de un pipeline en funcionamiento detecta final de flujo y error durante procesamamiento. Estos métodos a su vez realizan una invocación sobre los correspondiente métodos del EJB VsActionBean , pasándole el identificador del pipeline involucrado, que a su vez se corresponde con el de la sesión VS en el servidor de control y señalización.

-sendFileRecorded(): Este método será invocado desde el método terminateSessión(), cuando un pipeline recibiendo video sea parado. Este método pasará al EJB VsActionBean el fichero de video grabado y el identificador de sessión VS como parámetros en la invocación remota.

- **Integración de Liveserve**

El servidor Liveserve de la empresa Solaiemes esta desarrollado en lenguaje de programación Java y provee un API mediante el cual se pueden crear de forma sencilla sesiones para publicación y recepción de flujos tanto RTSP como RTMP.

Cuando se haya solicitado al servidor de video envío de video en vivo o recepción con publicación de video se creará de forma paralela, a través del API de Liveserve, una sesión de streaming. Para crear una sesión en Liveserve se requiere indicar los siguientes parámetros:

-Protocolo a emplear: RTMP, RTSP o ambos.
-Funcionamiento como servidor o cliente (publicación o recepción de flujo).
-Identificador de la sesión. En el caso de que funcione como servidor de video en streaming empleará este identificador para formar la url de tipo RTSP/RTMP donde se localiza el flujo de video.

rtsp:// nombre_host:7001/identificador_sesion
rtmp://nombre_host:5566/identificador_session

El host y el puerto son pasados por configuración. Los puertos por defecto si no se configuran son 5566 para rtmp y 7001 para rtsp.

-Url del flujo rtsp o rtmp al que conectarse, en el caso de que funcione como cliente**.**
-Host y puerto de la conexión RTP, desde el que recibirá el flujo RTP para servirlo en streaming, o hacia el que envíara el flujo de video recibido en streaming . En nuestro caso como Liveserve ha sido integrado como un módulo más del servidor de video, la conexión RTP sera interna.

Mediante las funciones start() y stop() del API de Liveserve se activará y desactivarán las sesiones en streaming de forma paralela a las sesiones generales VS del servidor de video.

Capítulo 5

Caso de uso: Videoblog.

En este capítulo se presenta una aplicación ejemplo como caso de uso del sistema implementado. El objetivo es dar una perspectiva general acerca de como desarrolladores de aplicaciones pueden hacer uso del API del Gateway para incluir en sus aplicaciones servicio Video Share.

La aplicación consiste en un videoblog, que permitirá publicar a modo de posts los videos grabados desde la cámara de un terminal móvil. Adicionalmente el videoblog incluirá un reproductor de video en tiempo real, de manera que se puede visualizar en vivo, a través de un navegador, el video capturado y emitido desde el móvil. A través de esta sencilla aplicación nos podremos dar una idea acerca de las posibilidades que ofrece el Gateway RCS-e VS.

El capítulo se divide en dos apartados, en el primero de ellos se trata de explicar cómo se ha llevado a cabo la implementación de la aplicación, mientras que en el segundo se expone detalladamente como es el funcionamiento de la aplicación desde la experiencia del usuario final.

5.1 Desarrollo de la aplicación videoblog.

Para la implementación de la aplicación videoblog, adicionalmente al API del Gateway RCS-e VS, se han empleado una serie de tecnologías y elementos orientados al desarrollo web. Aquí, sin entrar en detalle, nos limitaremos a listarlos comentando con qué objetivo se han empleado:

-**Servicio Blogger [65]:** es un servicio de Google que permite crear y publicar un blog en línea. Este servicio entre otras muchas funcionalidades cuenta con un diseñador de plantillas para personalizar tu blog y un editor de entradas que permite añadir texto, imágenes y videos sin necesidad de emplear programación. Para nuestra aplicación el primer paso consistirá en crear el blog usando el diseñador de plantillas

- **El API de datos de Blogger [66]**: es una API ofrecida por Google para la integración del servicio Blogger en aplicaciones desarrollada por terceros. Mediante este API se puede acceder a las entradas de un blog y se puede actualizar los contenidos de éste. Se hará uso de este API en su versión java, más concretamente se empleara la función createPost(), que nos permitirá añadir los videos grabados como entradas en el blog.

Para emplear createPost(), habrá que pasarle por parámetro el código HTML de la nueva entrada a postear, además de otros parámetros relacionados con la autenticación para acceso al blog. En nuestro caso el código de la entrada consiste en un sencillo iframe donde es embebido un reproductor flash, al cual se le pasa por parámetro la URL del fichero de video a reproducir.

-**Reproductores player.swf y SimpleFlashPlayer.swf [67]**: son dos reproductores flash de distribución libre, que pueden ser embebidos en una página web para la reproducción de video. El primero permite reproducir, entre otros, ficheros mp4 pasándole la URL HTTP donde se localiza éstos, mientras que el segundo se puede emplear para reproducir en streaming un flujo de video RTMP, pasándole como parámetro una URL de tipo RTMP actuará como cliente, conectándose a un servidor RTMP y reproduciendo el video en tiempo real.

-**Visor.jsp**: para la visualización en tiempo real del video se ha creado un JSP (Java Server Page) [68], que será embebido en nuestro blog como si de una entrada más de este se tratará. Implementado en Javascript [69], incluye un mecanismo de pushlet (Java Pushlet) [70] mediante el cual se establece un canal de eventos entre el browser y la aplicación (servidor web). El funcionamiento del JSP es el siguiente:

1. El contenido mostrado por defecto será un mensaje “NO LIVE VIDEO NOW”.

2. Cuando la aplicación videoblog haya establecido una sesión de video entrante en tiempo real con uno de sus usuarios, la aplicación generará un evento pushlet “start” que incluye la URL del flujo de video. El JSP sustituirá el mensaje por defecto por el reproductor SimpleFlashPlayer.swf y enlazará la URL, visualizándose así el flujo de video entrante en el visor del blog.

3. Cuando la sesión de video share termine, un nuevo evento “stop” será generado por la aplicación volviendo el JSP a mostrar el contenido por defecto.

En cuanto a la integración del servicio VS en la aplicación, en el siguiente diagrama se representa la interacción de la aplicación con el Gateway, mostrando como emplea las acciones y notificaciones del API.

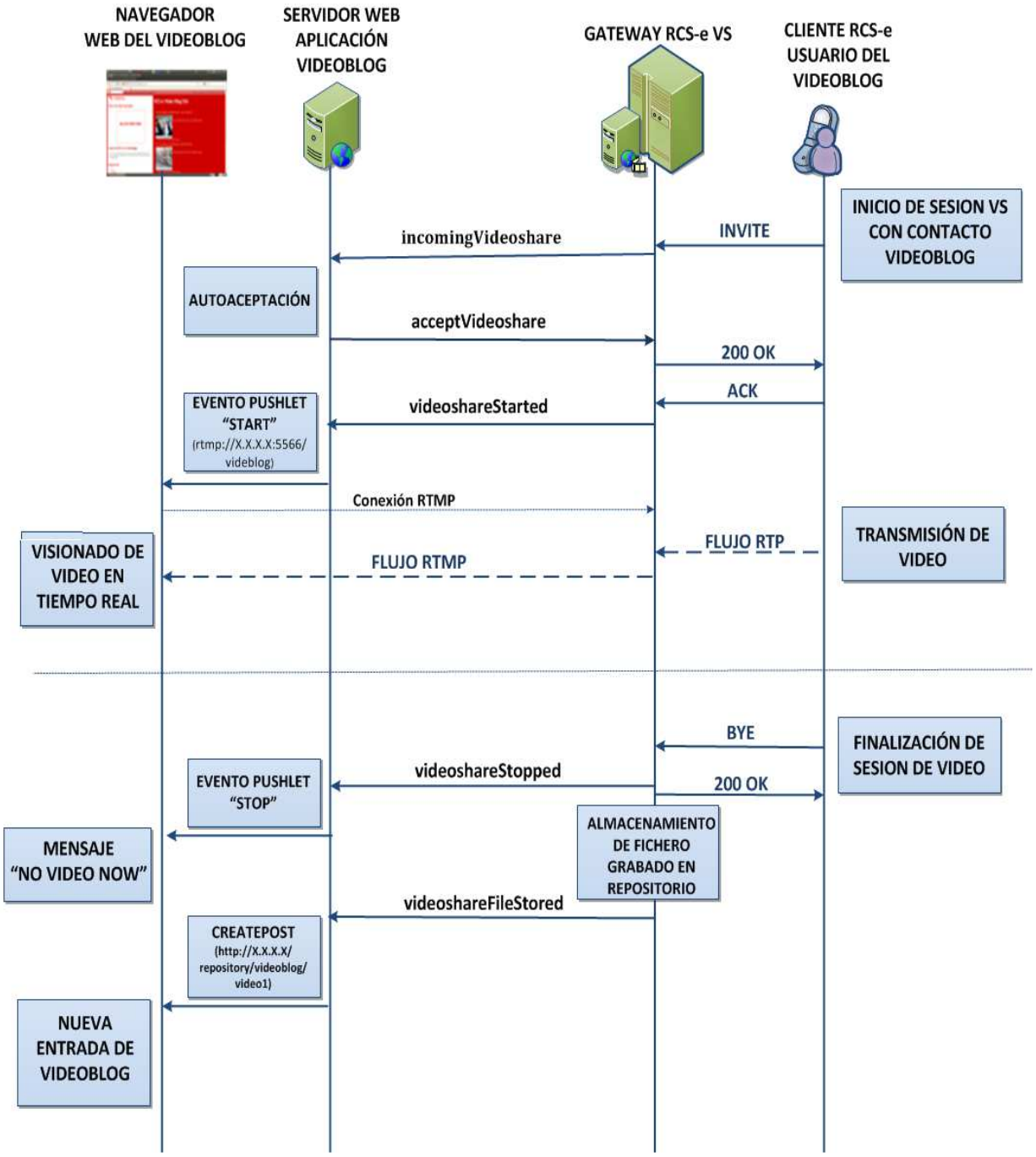


Figura 38.Integración del API Video Share en la aplicación Videoblog.

1. El usuario inicia una sesión VS con la aplicación Videoblog, que forma parte de su agenda, como si de un contacto más se tratará. El Gateway recibe un mensaje SIP INVITE, y notifica a la aplicación Videoblog que hay una sesión VS entrante para ella, indicándole la SIP URI del usuario originante.

2. La aplicación ante la notificación **incomingVideoshare**, simplemente se limitará a aceptar automáticamente la sesión de video sin comprobación alguna. Invocando la acción **acceptVideoshare** solicitará, a través de sus parámetros, que el video sea publicado usando RTMP y que el video sea grabado y almacenado en el repositorio.

3. El Gateway completará el establecimiento de sesión e indicará a la aplicación mediante la notificación **videoshareStarted**, que la transmisión de video se va a iniciar, aportándole como parámetro de la notificación la URL de tipo RTMP desde donde se emitirá el video en tiempo real.

4. La aplicación ante la notificación **videoshareStarted** generará un evento pushlet "start" que incluye la URL del flujo de video. Esto llevará a la visualización del video emitido desde el terminal móvil a través del elemento Visor.jsp embebido en el blog.

5. Cuando el usuario desde su terminal móvil finalice la sesión VS. El Gateway se lo notificará de forma inmediata a la aplicación, mediante la notificación **videoshareStopped,** y después se ocupará de llevar el video grabado durante la sesión hasta el repositorio.

6. Ante la notificación **videoshareStopped** la aplicación generará el evento pushlet "stop", pasándose a visualizar en el visor del blog un mensaje informativo acerca de que no hay emisión de video en ese momento.

7. Cuando el Gateway finalice el almacenamiento del video en el repositorio generará una nueva notificación, en este caso **videoshareFileStored,** que será mandada hacia la aplicación incluyendo la URL del repositorio donde se encuentra almacenado el fichero de video grabado.

8. La aplicación cuando reciba la notificación **videoshareFileStored**, generará un código HTML en el que se incluye un reproductor de video flash al que se le pasa por parámetro la localización del fichero de video grabado. Empleando la función createPost() del API de Blogger, el video aparecerá como una nueva entrada del blog, pudiéndose reproducir cuando se quiera desde la página web del videoblog.

Consideraciones adicionales:

-La aplicación requerirá aprovisionamiento previo en el sistema por parte del administrador del Gateway, éste introducirá el perfil de medias y el perfil SIP en base de datos, previo acuerdo con el desarrollador de la aplicación y con el operador del core. También será necesario que el desarrollador de la aplicación proporcione las URLs

donde se encuentran los descriptores WSDL de los web services, que emplea para recibir los mensajes SOAP de notificación.

-Por simplicidad, la aplicación implementada ha sido configurada en el Gateway en modo autoaceptación de contactos [ver apartado 3.1.7], de manera que cualquier usuario que introduzca en su agenda de contactos el número de teléfono a partir de la cual se conforma la SIP-URI o TEL-URI del videoblog [ver apartado 2.6.2], pasará a formar parte de la lista de usuarios de la aplicación. En este caso no será necesario emplear las acciones provistas para la gestión de lista de usuarios y no se tratarán las notificaciones acerca de cambios en ésta.

-Para el inicio de sesión en el core, mediante registro, se ha configurado la aplicación para que durante su despliegue en el servidor de aplicaciones se realice la invocación de la acción de registro de forma automática sin que sea necesario interacción alguna con un usuario. En este caso por simplicidad tampoco se dará tratamiento alguno a las notificaciones en cuanto a registro satisfactorio o fallo de registro.

La aplicación desarrollada en lenguaje de programación Java y desplegada sobre un servidor de aplicaciones JBoss, cuenta con cinco clases principales:

-**VideoShareReceiverWSImpl.java**: Esta clase implementa el web service sobre el que se recibirán las notificaciones del Gateway RCS-e VS. Se ha utilizado la herramienta JAX-WS para modelar las operaciones-notificaciones del Web Service como los métodos de la clase, éstos serán invocados desde el Gateway mediante mensajes SOAP encapsulados en peticiones HTTP.

-WSClient.java: Esta clase modela los clientes web services mediante los cuales se invoca las acciones del API del Gateway. Empleando también las librerías JAX-WS, incluye dos puntos de acceso (proxies), uno para invocar la operación register que forma parte del web services SessionWS y otro para la invocación de la operación acceptVideoshare del web services VideoshareWS del Gateway.

-**BloggerManager.java:** Esta clase emplea las librerías del API Data de Blogger para la generación y creación de las entradas del videoblog.

-**VideoSharePusher.jav**a: Esta clase emplea las librerias de Java Pushlet (nl.justobject.pushlet) y se encarga de generar los eventos a través de los que la aplicación se comunica con el Visor.jsp, encargado de generar el contenido web del videoblog para la visualización del video en vivo.

-Data.java: Los atributos estáticos de esta clase representan los distintos datos que la aplicación manejará durante el funcionamiento del videoblog:

Identificador del blog y password de acceso al mismo

Identificador de usuario con el que la aplicación accede al Gateway.

URLs de los descriptores WSDL de los servicios web del Gateway, para la generación y envío de mensajes SOAP con los que se invocara las acciones.

Localización de los archivos SWF (reproductores de videos).

URL donde se localiza el elemento Visor.jsp.

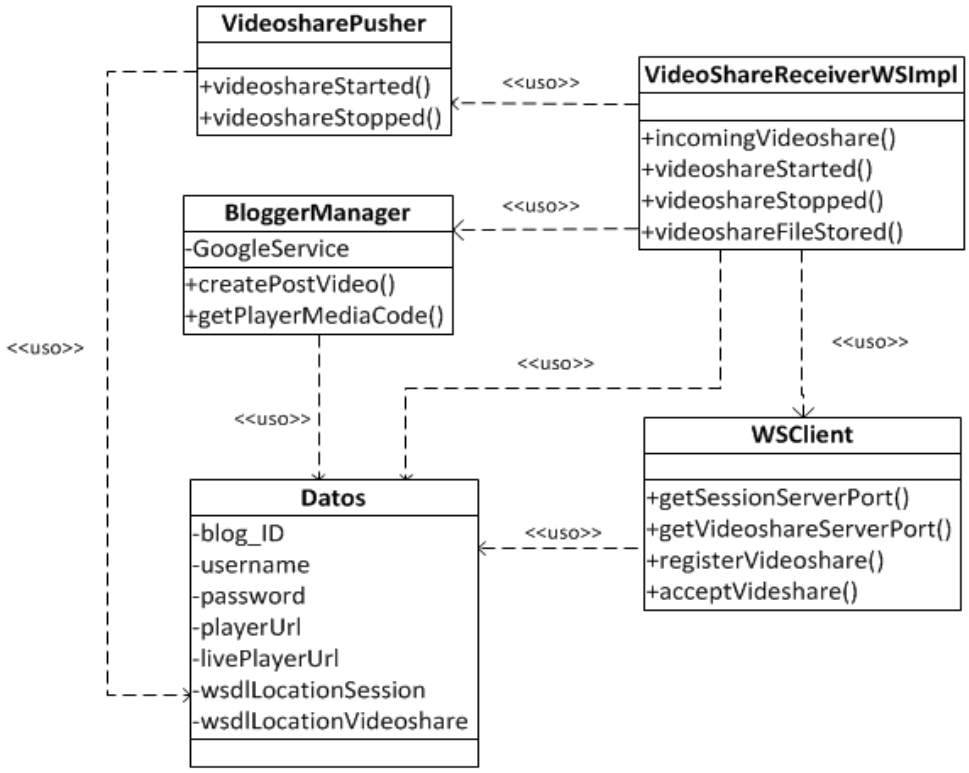


Figura 39. Diagrama de clases de la aplicación Videoblog

Todas estas clases será incluidas junto con el Visor.jsp, los ficheros SWF de los flash players y otros ficheros de configuración para el mecanismo pushlet, en un fichero WAR [47], que será desplegado en el contenedor web Tomcat que incluye el servidor de aplicaciones Jboss.

Como aplicación web, para el despliegue será también necesario incluir un fichero web.xml donde se configurarán aspectos como la localización del WSDL del Web Service implementado por la clase VideoShareReceiverWSImpl para la recepción de las notificaciones, la invocación de la acción register al desplegar, o el empleo del mecanismo de eventos pushlet.

5.2 Funcionamiento del videoblog.

En este apartado se presenta el funcionamiento del videoblog desde la perspectiva de los usuarios finales. Se distinguirán dos tipos de usuarios:

-Los usuarios que acceden a la aplicación a través de un cliente RCS-e con el objetivo de llevar el video capturado por la cámara de su móvil hasta el blog, mediante el establecimiento de una sesión VS.

-Los usuarios que acceden a la web del blog, a través de un navegador, para la visualización de los videos, ya sea en vivo o accediendo al histórico de videos emitidos que se presenta como entradas del blog.

A continuación se mostrará como es la experiencia de los usuarios empleando el servicio de videoblog, mediante imágenes de las interfaces de ambos, el cliente RCS-e del móvil y la web del videoblog,

Para esta demostración como cliente RCS-e se ha empleado un cliente de código abierto desarrollado por el laboratorio Orange para la plataforma Android, que cumple con los estándares marcados por RCS-e [71]. El cliente es instalado en el terminal móvil como cualquier otra aplicación Android. Para el caso de la interfaz web del videoblog, se ha realizado un diseño muy sencillo empleando el diseñador de plantillas del servicio Blogger. En la parte derecha de la página se mostrarán mediante una imagen "thumb" los videos posteados a los que se puede acceder directamente a través de los players embebidos en la página. La parte superior izquierda se ha reservado para el visor de video en vivo, que mostrará un mensaje informativo mientras no se esté recibiendo video.

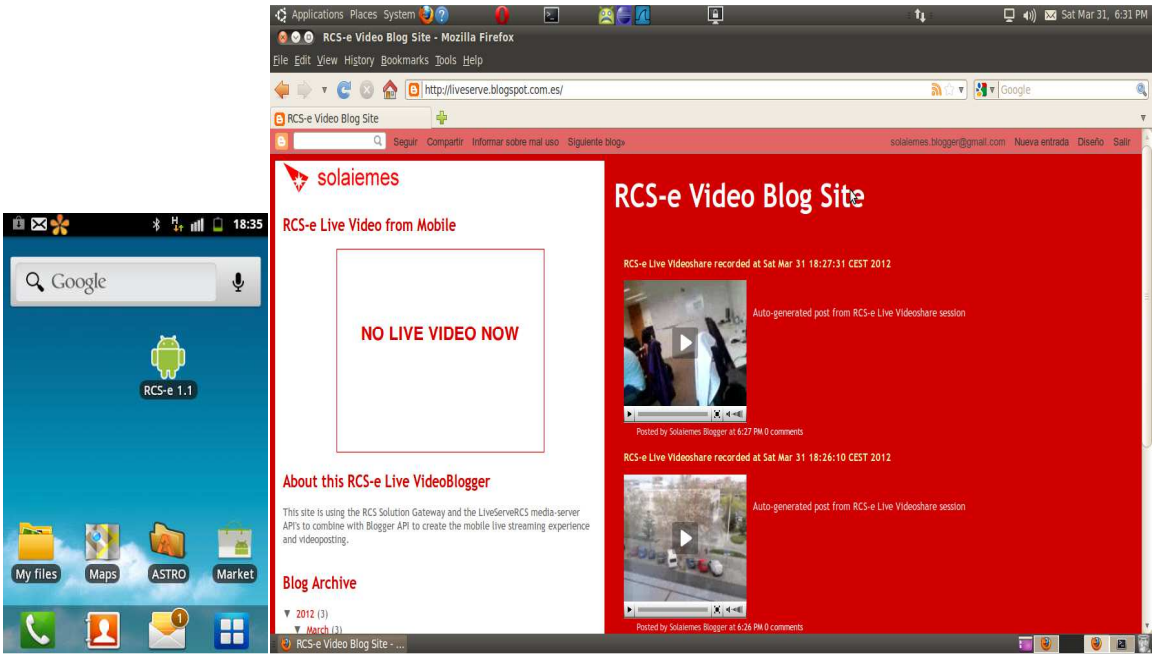


Figura 40. Cliente RCS-e e interfaz web del Videoblog.

El usuario desde su terminal móvil, cuando acceda al cliente RCS-e, se le mostrarán toda su lista de contactos (la agenda del móvil es integrada en el cliente RCS-e). Como si de un contacto RCS-e más se tratará, el servicio Videoblog aparecerá en esta lista a la que previamente se añadió introduciendo el número del teléfono que fue asignado al servicio por el operador y a partir del cual se forma la dirección SIP-URI o TEL-URI para servicios RCS-e.

Accediendo al contacto Videoblog se desplegarán un conjunto de opciones del servicio RCS-e, entre las que figura la posibilidad de ver en tiempo real las "capabilities" del contacto, es decir, las formas en las que podemos comunicarnos con él en ese momento. Para el caso del videoblog, se puede observar que tiene activada la opción de servicio Video Share.

Una vez comprobado que el servicio ésta activo, el usuario seleccionará la opción de "Rich call", habilitándose la posibilidad de realizar una llamada de voz previa, o directamente mandar una invitación Video Share al contacto Videoblog. Esta última será la opción elegida para empezar la transmisión de video hacia el blog.

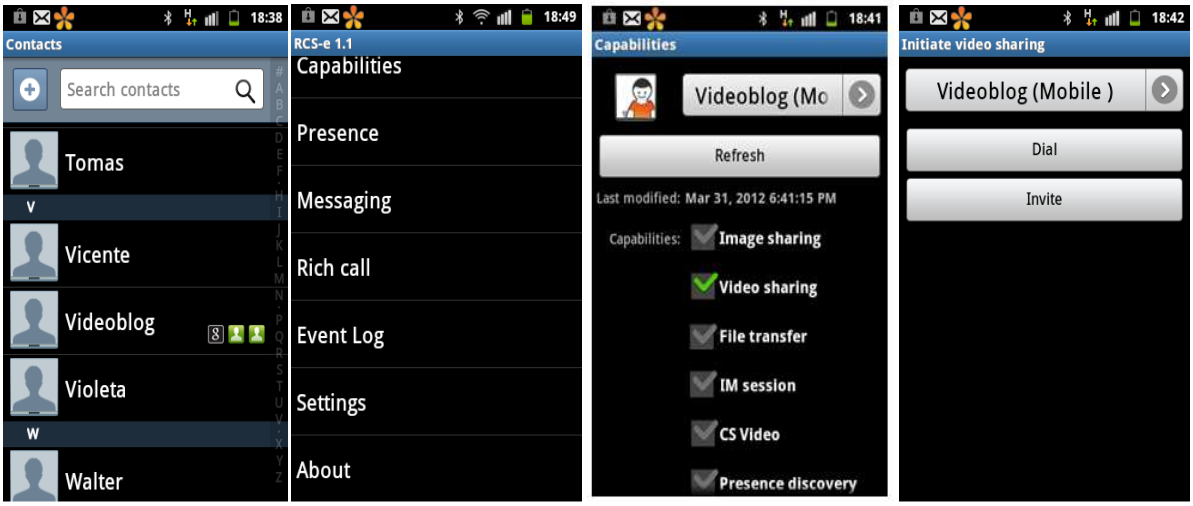


Figura 41. Cliente RCS-e. Acceso al servicio Videoblog.

La aplicación está configurada para la autoaceptación de toda sesión VS entrante por lo que inmediatamente comienza la transmisión del video capturado desde la cámara del terminal móvil. El usuario del cliente RCS-e podrá ver el video que está transmitiendo a través del visor que incorpora el cliente en la parte superior derecha de la pantalla del móvil. También se le mostrará activado un botón "Stop", para el cierre de la sesión cuando lo desee.

Del lado del videoblog un reproductor flash ha sustituido el mensaje que el visor muestra por defecto y el usuario ya puede visualizar el video que en ese mismo momento está siendo emitido por el terminal móvil.

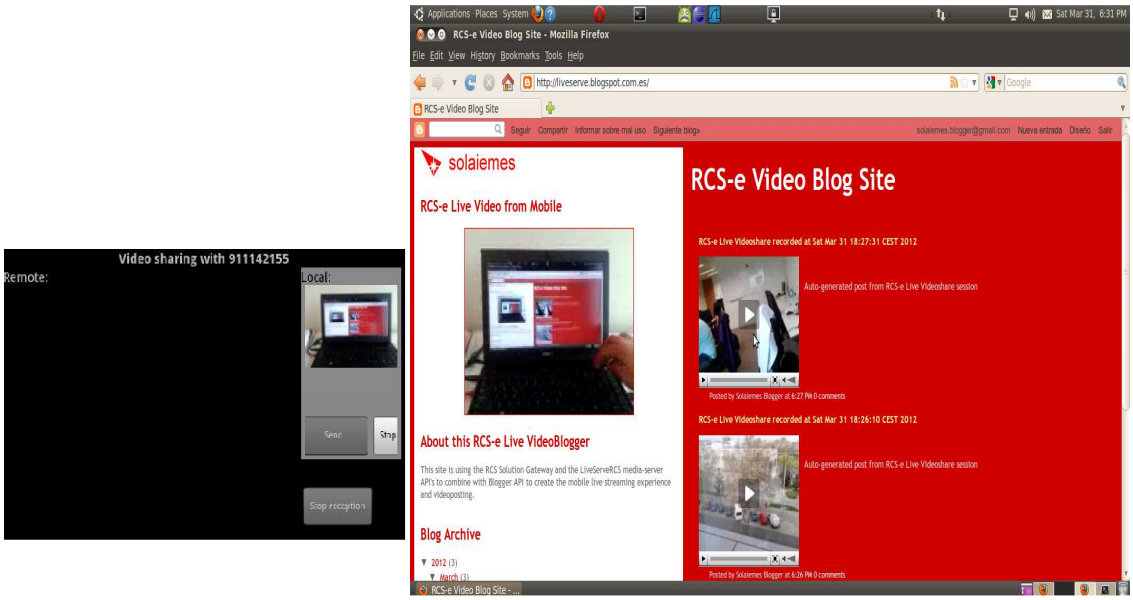


Figura 42. Transmisión de video en vivo hacia el Videoblog.

Cuando el usuario emisor del video desee cerrar la transmisión pulsará el botón “Stop”, la transmisión finalizará inmediatamente siendo así anunciado por el Videoblog, que volverá a mostrar el mensaje “NO LIVE VIDEO NOW”.

Al cabo de un par de segundo, a la derecha aparecerá un reproductor de video mostrando una imagen thumb del video que se acaba de recibir. El video emitido fue grabado y se encuentra enlazado desde el repositorio de video al reproductor flash, de forma que el usuario puede reproducir el archivo cuando lo desee.

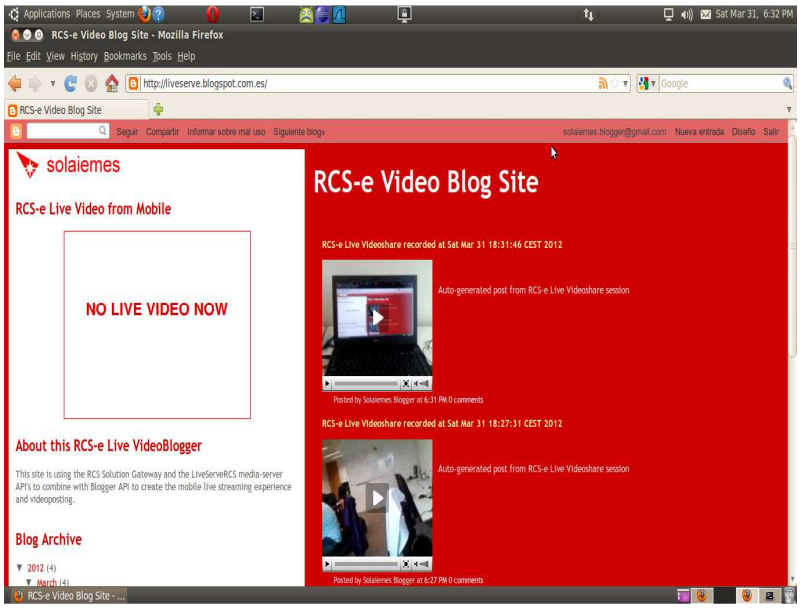


Figura 43. Posteo de un nuevo video en el Videoblog.

Capítulo 6

Planificación y presupuesto

Este capítulo está dedicado a la presentación de la planificación y el presupuesto del proyecto realizado. En primer lugar veremos la planificación, donde el trabajo realizado se ha dividido en fases, la mayor parte reflejadas a lo largo de este documento, que a su vez se componen de una sucesión de tareas comúnmente requeridas para un proyecto de desarrollo software. En el segundo capítulo se ha llevado a cabo el cómputo de costes asociados al proyecto, se ha especificado el personal empleado y los costes de equipamiento requeridos para finalmente dar el presupuesto total.

6.1 Planificación del proyecto.

Para llevar a cabo la planificación se ha empleado la herramienta Microsoft Project [72], mediante la cual se ha generado un diagrama de Gantt. Esta herramienta es de uso muy extendido en gestión de proyectos y permite visualizar fácilmente la relación entre tarea y tiempo empleado, así como la secuenciación de las mismas y sus dependencias.

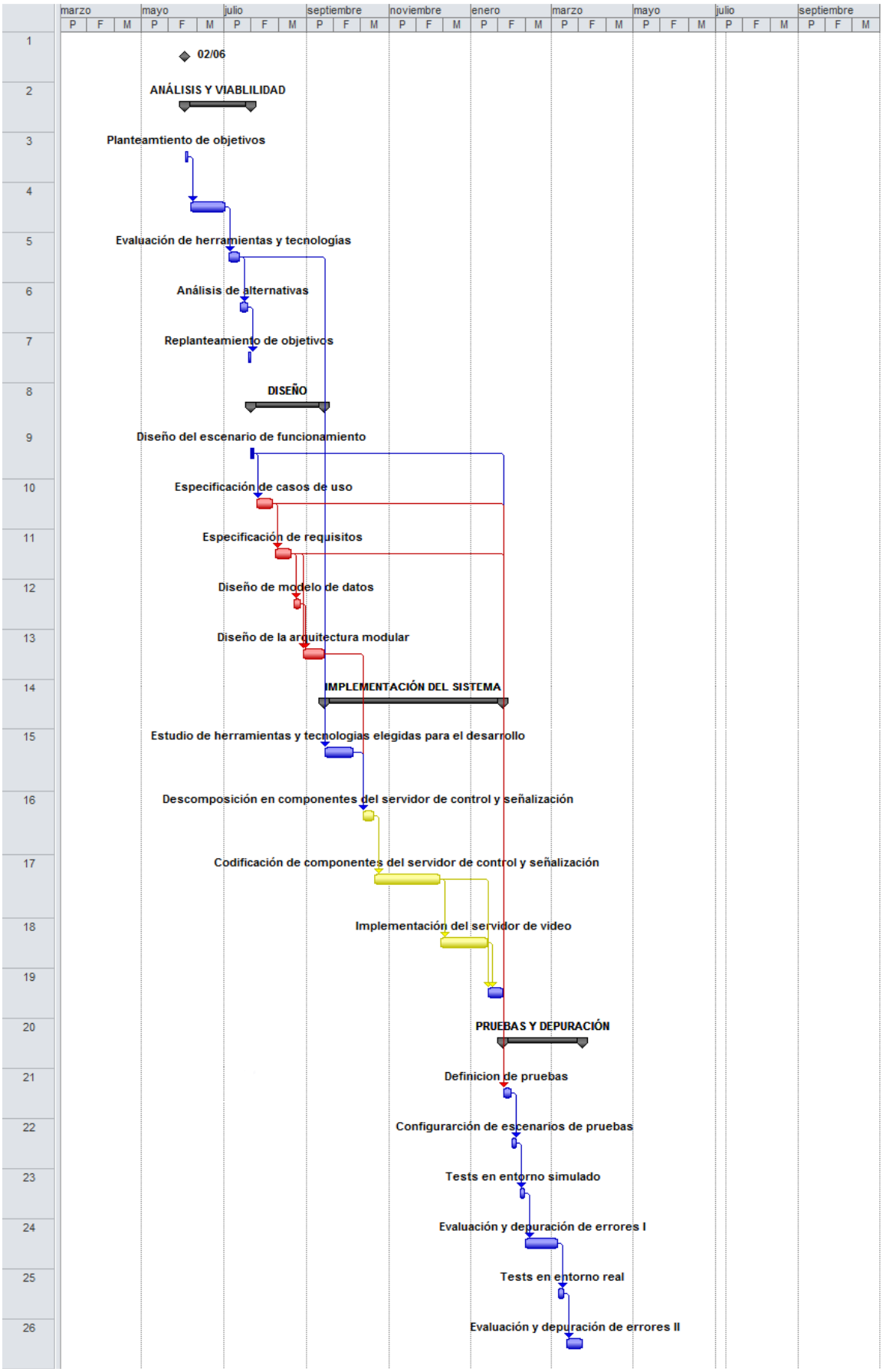
En la elaboración del diagrama se ha tenido en cuenta jornadas laborales completas. Considerando que el tiempo de desarrollo del proyecto no siempre fue continuado, se ha realizado ciertos ajustes, por lo que esta planificación debería ser tomada como una estimación

A continuación se presenta una tabla donde se incluye todas las tareas en orden cronológico junto con sus tiempos de ejecución. A partir de dicha tabla se genera el diagrama de Gantt que será presentado posteriormente.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio del proyecto	0 días	jue 02/06/11	jue 02/06/11	
2	**ANÁLISIS Y VIABLILIDAD**	**35 días**	**jue 02/06/11**	**mié 20/07/11**	
3	Planteamtiento de objetivos	2 días	jue 02/06/11	vie 03/06/11	
4	Estudio del estado del arte	20 días	lun 06/06/11	vie 01/07/11	3
5	Evaluación de herramientas y tecnologías	7 días	lun 04/07/11	mar 12/07/11	4
6	Análisis de alternativas	4 días	mié 13/07/11	lun 18/07/11	5
7	Replanteamiento de objetivos	2 días	mar 19/07/11	mié 20/07/11	6
8	**DISEÑO**	**39 días**	**jue 21/07/11**	**mar 13/09/11**	
9	Diseño del escenario de funcionamiento	2 días	jue 21/07/11	vie 22/07/11	
10	Especificación de casos de uso	10 días	lun 25/07/11	vie 05/08/11	9
11	Especificación de requisitos	10 días	lun 08/08/11	vie 19/08/11	10
12	Diseño de modelo de datos	5 días	lun 22/08/11	vie 26/08/11	11
13	Diseño de la arquitectura modular	12 días	lun 29/08/11	mar 13/09/11	12;11
14	**IMPLEMENTACIÓN DEL SISTEMA**	**95 días**	**mié 14/09/11**	**mar 24/01/12**	
15	Estudio de herramientas y tecnologias elegidas para el desarrollo	15 días	mié 14/09/11	mar 04/10/11	5
16	Descomposición en componentes del servidor de control y señalización	7 días	mié 12/10/11	jue 20/10/11	15;13
17	Codificación de componentes del servidor de control y señalización	35 días	vie 21/10/11	jue 08/12/11	16
18	Implementación del servidor de video	25 días	vie 09/12/11	jue 12/01/12	17
19	Integración del sistema	8 días	vie 13/01/12	mar 24/01/12	17;18
20	**PRUEBAS Y DEPURACIÓN**	**43 días**	**mié 25/01/12**	**vie 23/03/12**	
21	Definicion de pruebas	4 días	mié 25/01/12	lun 30/01/12	9;10;11
22	Configurarción de escenarios de pruebas	4 días	mar 31/01/12	vie 03/02/12	21
23	Tests en entorno simulado	4 días	lun 06/02/12	jue 09/02/12	22
24	Evaluación y depuración de errores I	17 días	vie 10/02/12	lun 05/03/12	23

24	Evaluación y depuración de errores I	17 días	vie 10/02/12	lun 05/03/12	23
25	Tests en entorno real	4 días	mar 06/03/12	vie 09/03/12	24
26	Evaluación y depuración de errores II	10 días	lun 12/03/12	vie 23/03/12	25
27	**DESARROLLO DE APLICACIÓN VIDEOBLOG**	**16 días**	**lun 26/03/12**	**lun 16/04/12**	
28	Evaluación y estudio de herramientas y tecnologias web	5 días	lun 26/03/12	vie 30/03/12	
29	Diseño de la aplicación	3 días	lun 02/04/12	mié 04/04/12	28
30	Implementación de la aplicación	5 días	jue 05/04/12	mié 11/04/12	29
31	Testeo y depuración de la aplicacion	3 días	jue 12/04/12	lun 16/04/12	30
32	**DOCUMENTACIÓN**	**71 días**	**mar 17/04/12**	**mar 24/07/12**	
33	Planificación de la memoria	5 días	mar 17/04/12	lun 23/04/12	
34	Recopilación de información y documentos	15 días	mar 24/04/12	lun 14/05/12	33

Tabla 14. Fases y tareas del proyecto.


marzo
mayo
julio
septiembre
noviembre
enero
marzo
mayo
julio
septiembre
P F M
1
02/06
2
ANÁLISIS Y VIABLILIDAD
3
Planteamtiento de objetivos
4
5
Evaluación de herramientas y tecnologías
6
Análisis de alternativas
7
Replanteamiento de objetivos
8
DISEÑO
9
Diseño del escenario de funcionamiento
10
Especificación de casos de uso
11
Especificación de requisitos
12
Diseño de modelo de datos
13
Diseño de la arquitectura modular
14
IMPLEMENTACIÓN DEL SISTEMA
15
Estudio de herramientas y tecnologías elegidas para el desarrollo
16
Descomposición en componentes del servidor de control y señalización
17
Codificación de componentes del servidor de control y señalización
18
Implementación del servidor de video
19
20
PRUEBAS Y DEPURACIÓN
21
Definicion de pruebas
22
Configurarción de escenarios de pruebas
23
Tests en entorno simulado
24
Evaluación y depuración de errores I
25
Tests en entorno real
26
Evaluación y depuración de errores II

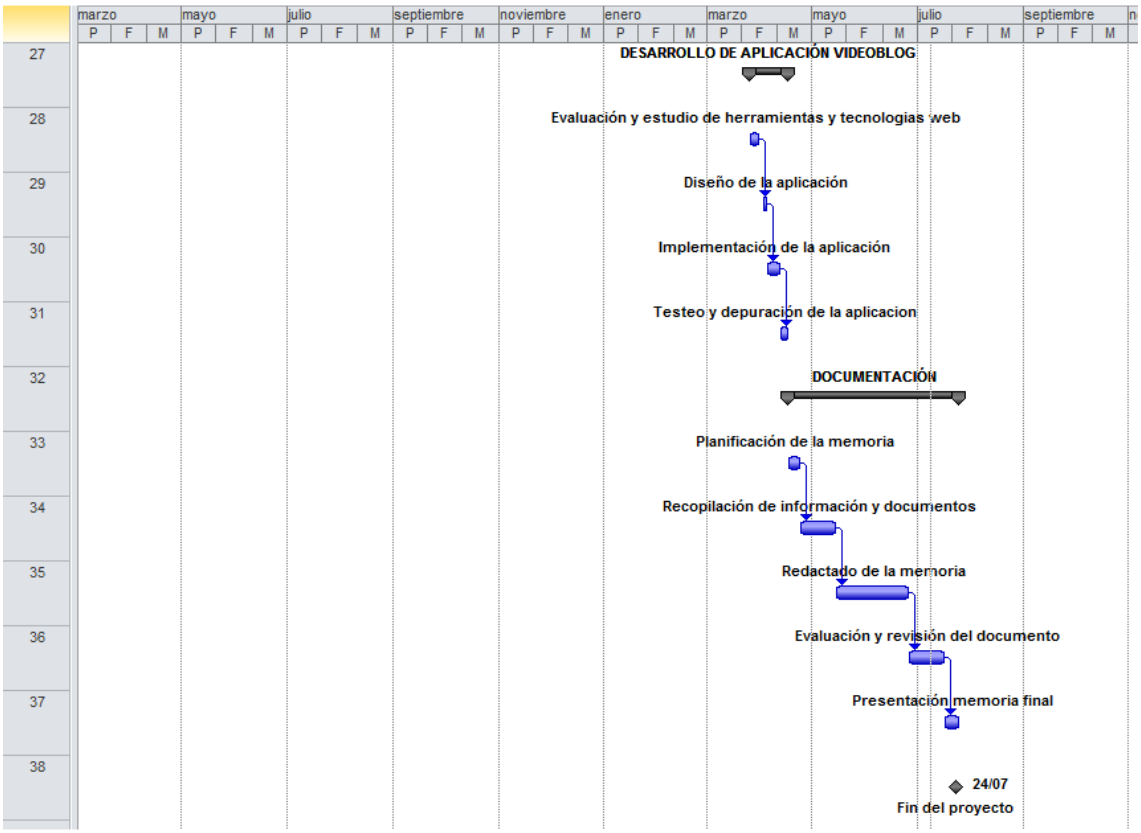


*Figura 44. Diagrama de Ga*ntt

6.2 Presupuesto del proyecto.

Para la realización del presupuesto se ha empleado la plantilla proporcionada por la Escuela Politécnica Superior de la Universidad Carlos III de Madrid. Mediante esta plantilla se ha realizado el desglose de costes y se ha calculado el presupuesto total, que asciende a la cantidad de **66.108 Euros**.

Hay que tener en cuenta que en materia de personal se ha introducido, adicionalmente al alumno, al Jefe de Proyecto. La dedicación de esta figura ha sido a tiempo parcial, realizando funciones de supervisión en diferentes fases del proyecto, se estima que el tiempo dedicado ha sido de 4 meses. En cuanto a los equipos empleados, adicionalmente al ordenador personal en el cual se ha realizado el desarrollo del proyecto, se han empleado dos terminales móviles y un servidor, de forma compartida, durante algunas fases del proyecto, donde era necesario recrear el escenario de funcionamiento.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

JESUS AVILES AVILES

2.- Departamento:

INGENIERÍA TELEMÁTICA

3.- Descripción del Proyecto:

- Titulo: RCS-e VIDEOSHARE GATEWAY
- Duración (meses): 14
- Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

70.000,00 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
				0,00
Valencia, Luis	Ingeniero Senior	4	4.289,54	17.158,16
Avilés Avilés, Jesus	Ingeniero	14	2.694,39	37.721,46
				0,00
				0,00
	Hombres mes	**18**	**Total**	54.879,62

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Portátil Dell Vostro	589,00	100	14	60	137,43
Servidor Dell PowerEdge T320	879,00	50	6	60	43,95
Movil Samsung Galaxy Ace	196,00	50	8	60	13,07
Movil HTC	238,00	50	8	60	15,87
				Total	210,32

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	54.880
Amortización	210
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	11.018
Total	66.108

Capítulo 7

Conclusiones y trabajos futuros

En este último capítulo, primero se presentan la principales conclusiones extraidas del proyecto realizado y posteriormente se exponen algunas posibles líneas de trabajo futuro.

7.1 Conclusiones.

Este proyecto tenía como objetivo principal el diseño y desarrollo de un sistema Gateway, mediante el cual, el servicio VS definido por RCS-e para la compartición de video entre dos terminales móviles es abierto a desarrolladores (terceras partes) para la creación de casos de uso donde el servicio VS es integrado con soluciones web y servicios de contenidos. Como resultado final se ha obtenido un prototipo funcional, cuya validación se ha llevado a cabo a través de un caso de uso concreto, la aplicación videoblog, obteniendo las siguientes conclusiones de carácter general:

- El sistema propuesto puede resultar ventajoso para las operadoras, ya que pueden incorporar servicios de valor añadido sin introducir cambios en su infraestructura (servidores de aplicaciones y otros elementes IMS) y sin perder el control del servicio (todas las sesiones VS pasan por el core). Las aplicaciones acceden al core del operador a través de clientes virtuales, empleando la interfaz UNI, del mismo modo a como lo hacen los clientes RCS-e embebidos en los terminales móvil.

- El usuario final puede acceder a las aplicaciones de video desde su agenda del móvil. Todas las aplicaciones en una misma interfaz, sin necesidad de instalar nuevo software para cada aplicación. Las aplicaciones son añadidas a la agenda como si de un contacto más se tratase.

- En cuanto a desarrolladores de aplicaciones, este sistema, a través de su API de Web Services, permite una fácil integración del servicio. Empleando herramientas y tecnologías comúnmente usadas en el desarrollo web pueden crear multitud de casos de uso para video e incorporar VS a los ya existentes.

Otras conclusiones extraídas durante el diseño y la implementación del sistema son:

-El papel de los estándares y especificaciones es fundamental para la interoperabilidad de los servicios, la evolución de éstos y su aceptación. El diseño funcional del Gateway se basa en la especificación RCS-e, donde se definen los protocolos y mecanismos de funcionamiento del servicio VS. Además estándares web como HTTP o SOAP son empleados para llevar el servicio hasta las aplicaciones.

-Un buen diseño de la arquitectura, en un sistema de estas características, donde diversas funcionalidades, tecnologías y protocolos se conjugan, resulta vital para poder abordar la implementación del mismo. Se ha realizado una minuciosa descomposición de módulos funcionales y componentes software, mediante la que se facilita la adaptación a cambios, la reusabilidad del software, y la introducción de nuevas funcionalidades.

-Para la viabilidad de un proyecto como éste, con altas pretensiones y pocos recursos, resulta determinante la existencia de tecnología y herramientas de código abierto que sean consistentes y ofrezcan altas prestaciones. En nuestro caso Gstreamer, Mobicents SIP Servlets y JEE, constituyen la base tecnológica sobre la se ha implementado el sistema.

-La plataforma de programación JEE, ha facilitado sobremanera el desarrollo del servidor de control y señalización del sistema. Su modelo de componentes y sus numerosas APIs han permitido gestionar y procesar de manera sencilla pero eficiente la información asociada a sesiones, aplicaciones, y usuarios de las mismas. Además, mediante sus APIs para la implementación de Web Services, se ha creado el API del sistema a través del que las aplicaciones acceden al servicio VS.

-Gstreamer se presenta como una herramienta ya consolidada que permite el desarrollo de aplicaciones audiovisuales potentes. En nuestro caso ha sido empleada para la implementación del servidor de video, mediante el que se realiza el envio y recepción de video RTP para cada una de las sesiones VS establecidas entre usuario y aplicación.

-El servidor Mobicents SipServlets, plataforma elegida para el depliegue de nuestro sistema, constituye la piedra angular del Gateway, ya que ha permitido una convergencia natural pero consistente entre SIP y WEB.

7.2 Trabajos futuros.

El sistema desarrollado se trata de un prototipo funcional y como tal, está abierto a cuantiosas mejoras y son muchas las líneas de trabajo futuro. Aquí se exponen algunas de las que se han considerado más importantes:

- **Integración de Video-Share y llamada de voz**.

En este proyecto se ha seguido la premisa de independencia entre llamada de voz y sesión de video [ver apartado 3.1.6]. Aun así, con el objetivo de que el Gateway pueda interoperar con cualquier cliente RCS-e y aumentar sus posibilidades en cuanto a capacidad de servicio, se ha planteado la posibilidad de incorporar en el futuro un nuevo elemento al sistema, consistente en un gateway implementado sobre tecnología Asterisk [73] que permita la "traducción" de llamadas de voz tradicionales al dominio SIP y RTP, de forma que puedan ser gestionadas internamente por el RCS-e VS Gateway a modo de audio para el video.

- **Desarrollo de una aplicación para aprovisionamiento de aplicaciones.**

Cuando se da de alta a una aplicación en el sistema, es necesario introducir la información SIP asociada a la aplicación, información acerca de los parámetros de video que soporta, así como las URL para la recepción de notificaciones. Todos estos valores son introducidos mediante scripts SQL en base de datos. Resultaría muy interesante implementar una aplicación para la administración del Gateway que facilitara esta tarea.

- **Interoperabilidad con entornos reales**.

Durante el desarrollo del sistema se han hecho pruebas con el cliente RCS-e de los laboratorios Orange, aún en fase de desarrollo, y se ha empleado el sistema OpenSer [74] como core RCS/IMS de prueba. Aunque se ha realizado pruebas satisfactorias con el core en producción de Vodafone y su cliente RCS-e (bajo la marca Joyn) [75], el camino por recorrer respecto a la interoperabilidad del sistema en entornos reales es muy grande.

- **Escalabilidad del sistema.**

Teniendo en cuenta que el sistema debería dar soporte al mayor número de aplicaciones posibles y que éstas pueden ser de uso masivo (gran número de usuarios), la escalabilidad del sistema será un factor determinante a la hora de poder dar uso comercial a la plataforma. En fases posteriores será necesario hacer una evaluación

acerca de la escalabilidad del sistema y ver cual es el comportamiento de éste ante pruebas de carga.

- **Integración de servicios RCS-e bajo una misma plataforma.**

En paralelo a este proyecto, en la empresa Solaiemes se están desarrollando proyectos similares al que aquí se ha abordado, donde el servicio IM/chat y el servicio de transferencias de ficheros de RCS-e son también abiertos al desarrollo de aplicaciones. En el futuro el servicio VS debería ser integrado junto a estos servicios en una misma plataforma, con el objetivo de llevarlos de forma conjunta hasta las aplicaciones, creándose casos de uso más atractivos para el usuario.

Glosario de Acrónimos

API	Application Programming Interface
ACK	Acknowledgement
AS	Application Server
B2BUA	Back to Back User Agent
CRM	Customer Relationship Management
CS	Call Switch
DAO	Data Access Object
DNS	Domain Name System
EJB	Enterprise Java Beans
GSMA	Global System for Mobile communication Association
HTML	HiperText Markup Language
HTTP	HyperText Transfer Protocol
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IETF	Internet Engineering Task Force
IP	Internet Protocol
JAX-WS	Java API XML Web Service
JDBC	Java DataBase Connectivity
JEE	Java Enterprise Edition
JSP	Java Server Page
JPA	Java Persistence API
MRFC	Multimedia Resource Function Controller

MRFP Multimedia Resource Function Processor

MSS Mobicents Sip Servlets

OMA Open Mobile Alliance

PS Packet Switch

P2P Peer to Peer

QoS Quality of Service

RCS Rich Communication Suite

RCS-e Rich Communication Suite enhanced

RFC Request For Comments

RMI Remote Method Invocation

RR Receiver Report

RTCP Real-time Transport Control Protocol

RTMP Real Time Messaging Protocol

RTP Real-time Transport Protocol

RTSP Real Time Streaming Protocol

SDP Session Description Protocol

SIP *Session* Initiation Protocol

SOAP Simple Object Access Protocol

SQL Structured Query Language

SR Sender Report

SV Servidor de Video

TCP Transport Control Protocol

TLS Transport Layer Security

UGC User Generated Contents

UA User Agent

UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UNI	User Network Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VS	Video Share
WS	Web Service
WSDL	Web Service Description Language
XML	eXtensible Markup Language

Referencias

[1] Rich Communication. GSMA [www.gsmworld.com].

[2] GSMA**:** Global System for Mobile Communications Association [www.gsm.org].

[3] RCS-e Advanced Communication Version 1.1. Javier Arenzana, Philip Carter, Oscar Gallego, Sergio Garcia Murillo y otros.

[4] Rich Communication Suite Release 1 Functional Description Version 1.2

[5] Rich Communication Suite Release 2 Functional Description Version 2.0.

[6] OMA-PRESENCE-SIMPLE [www.openmobilealliance.org].

[7] Video Share Service Definition. GSMA Document SE.41.

[8] Video Share Interoperability Specification. GSMA Document IR.74.

[9] Video Share Phase 2 Interoperability Specifications. GSMA Document IR.84.

[10] The IMS: IP Multimedia Concepts and Services in the Mobile Domain. Mika Poikselka, Georg Mayer, Hisham Khartabil, Aki Niemi.

[11] The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds. Gonzalo Camarillo, Miguel-Ángel García-Martín

[12] 3GPP: 3rd Generation Partnership Project [www.3gpp.org].

[13] IETF: Internet Engineering Task Force [www.ietf.org].

[14] DIAMETER Base Protocol (RFC 3588). P.Calhoun, J.Loghney, E. Guttman, G. Zoon y J. Arkko.

[15] DNS: Domain Names –Concepts and Facilities (RFC 1034). P. Mockapetris.

[16] CAMEL: Customized Applications for Mobile Network Enhanced Logic, 3GPP TS 23.278 V7.1.0.

[17] OSA: Open Service Architecture. 3GPP TR 23.927.

[18] SCTP: An Introduction to the Stream Control Transmission Protocol (RFC 3286). L. Ong y J.Yokum.

[19] MTP: Functional description of the message transfer part of Signalling System No 7. ITU-T Recommendation Q.701

[20] ISUP: Validation and compatibility for ISUP'92 and Q.767 protocols. ITU-T Recommendation Q.784.1.

[21] H.248: Gateway control protocol. Version 3. ITU-T Recommendation H.248.1.

[22] G.711: Pulse code modulation (PCM) of voice frequencies. ITU-T Recommendation G.711.

[23] SIP: Session Initiation Protocol (RFC 3261). J. Rosenberg, H.Shulzrinne, G. Camarillo, A. Jhonston y otros.

[24] SDP: Session Description Protocol (RFC 4566). M. Handley, V. Jacobson y C. Perkins.

[25] Session Description Protocol (SDP) Offer/Answer Examples (RFC 4317). A. Jhonston y R. Sparks

[26] RTP: A Transport Protocol for Real-Time Applications (RFC 3550). H. Schulzrinne, S. Casner, R. Frederick y V. Jacobson.

[27] UDP: User Datagram Protocol. (RFC 768). J. Postel.

[28] Symmetric RTP / RTCP Control Protocol (RTCP) (RFC 4961). D. Wing.

[29] NTP: Network Time Protocol. Specification, Implementation and Analysis. (RFC 1305). David L. Mills.

[30] Introduction of GIBA procedures. 3GPP (Ericsson/Sean) C-082128.

[31] H.264: Advanced video coding for generic audiovisual service. ITU-T Recommendation H.264.

[32] Mpeg Industry Forum [http://www.m4if.org/mpeg4/].

[33] H.263: Video coding for low bit rate communication. ITU-T Recommendation H.263.

[34] RTP payload format for MPEG-4 Audio/Visual streams (RFC 3016). Y. Kikuchi, T. Nomura, S. Fkinaga, Y. Matsui y Y. Kimata.

[35] RTP Payload Format for H.263 Video Streams (RFC 2190). C. Zhu.

[36] RTP Payload Format for H.264 Video Streams (RFC 3984). S. Wagener, M. M. Hannuksela, T. Stockhammer, M. Westerlund y D. Singer.

[37] RTSP: Real Time Streaming Protocol (RFC 2326). H. Schulzrinne, A. Raop y R. Lanphier.

[38] RTP Profile for Audio and Video Conference with Minimal Control (RFC 3551). H. Schulzrinne y C. Casner.

[39] An Extension to HTTP: Digest Access Authentication (RFC 2617). J. Franks, P. Hallman-Bakker, J. Hostetler, S Lawrence y otros.

[40] HTTP- Hypertext Transfer Protocol. W3C [http://www.w3.org/Protocols/].

[41] Solaiemes S.L. [http://www.solaiemes.com/].

[42] Apache Jackrabbit [http://jackrabbit.apache.org/].

[43] JSR 170: Content Repository for Java technology API.

[44] RTMP Specification 1.0 [http://www.adobe.com].

[45] The Java EE 5 Tutorial. [http://docs.oracle.com/javaee/5/tutorial/doc/].

[46] JBoss at Work: A Practical Guide. Tom Marrs y Scott Davis

[47] Proyecto Tomcat [http://tomcat.apache.org/].

[48] Tutorial PostgreSQL. Equipo de desarrollo postgresSQL

[49] Enterprise Java Beans 3.0 Especificación [http://java.sun.com/products/ejb/docs]

[50] SOAP Version 1.2 Part 0 [http://www.w3.org].

[51] Guía breve de las tecnologías XML [http://www.w3c.es/Divulgacion].

[52] The Java Remote Method Invocation (RMI) Tutorial. [http://docs.oracle.com/javase/tutorial/rmi/]

[53] CORBA website [http://www.corba.org/].

[54] Java EE Techologies-Database [http://www.oracle.com/technetwork/java/javase/tech/].

[55] Estado del Arte: Servicios Web. Carlos Andrés Morales Machuca.

[56] Especificación SIP Servlet v1.1 (JSR289). Mihir Kulkarni y Yanni Cosmodupoulus

[57] Mobicents website [http://www.mobicents.org/].

[58] SIP Servlets Server User Guide. Douglas Silas, Jean Deruelle y otros.

[59] GStreamer Application Development Manual (0.11.91.1). Wim Taymans, Steve Baker, Andy Wingo y otros.

[60] Gstreamer-Java. Java interface to the gstreamer framework.

[http://code.google.com/p/gstreamer-java/].

[61] Java Native Access (JNA) website [https://github.com/twall/jna#readme]

[62] Cutnpaste. Blog sobre tecnología y desarrollo software [http://cutnpaste.org/]

[63] JAIN-SDP (JSR 141 release)

[64] JAIN-SIP-1.2 (JSR-32 maintenance release)

[65] Blogger Buzz. The Official Buzz from Blogger at Google. [http://buzz.blogger.com/]

[66] APIs de Datos de Google. [https://developers.google.com/gdata/]

[67] Embedding Flash. Longtail community. [http://www.longtailvideo.com/support/jw-player/13/embedding-flash]

[68] JavaServer Pages Technology. Oracle [http://www.oracle.com/technetwork/java/javaee/jsp/index.html]

[69] Manual de JavaScript. José Antonio Rodríguez

[70] Pushlet website [http://www.pushlets.com/]

[71] Android-rcs-ims-stack. RCS-e stack for Android platform [http://code.google.com/p/android-rcs-ims-stack/]

[72] Microsoft Office Project 2003 Step by Step. Carl Chatfiel and Tmothy Johnson

[73] Asterisk. The Open Source Telephony Projects. [www.asterisk.org]

[74] OpenSer website [www.opensips.org]

[75] Vodafone: Joyn versión Beta, una nueva forma de comunicarte. [http://www.vodafone.es/apps-y-descargas/es/servicios-para-smartphones/joyn/]